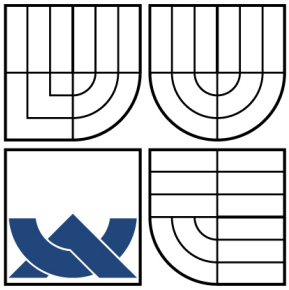
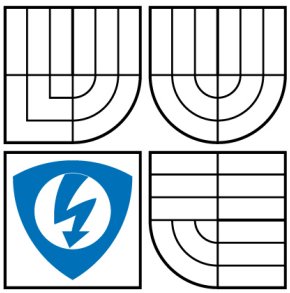


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

SOFTWAROVÁ PODPORA VÝUKY KRYPTOSYSTÉMŮ ZALOŽENÝCH NA PROBLÉMU FAKTORIZACE VELKÝCH ČÍSEL

SOFTWARE SUPPORT OF EDUCATION IN CRYPTOGRAPHY BASED ON INTEGER FACTORIZATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. PETR VYCHODIL
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. KAREL BURDA, CSc.
SUPERVISOR

BRNO 2009

ABSTRAKT

Tato diplomová práce se zabývá softwarovou podporou výuky asymetrických šifrovacích algoritmů, založených na problematice faktorizace velkých čísel. Byl vytvořen vzorový program, který umožňuje provádět operace spojené se šifrováním a dešifrováním s interaktivním ovládáním, s jehož pomocí lze jednoduchým způsobem pochopit princip šifrovací metody RSA.

V kapitolách 1 a 2 je rozebrána problematika šifrovacích algoritmů všeobecně. Kapitoly 3 - 5 se již podrobně věnují problematice šifrovacího algoritmu RSA, principům získání, správy a použití šifrovacích klíčů. Kapitola 5 popisuje možnosti zvolení vhodné technologie pro vytvoření konečného softwarového produktu, která by umožňovala vhodným způsobem prezentovat vlastnosti tohoto rozšířeného šifrovacího algoritmu RSA.

Konečným softwarovým produktem je java aplet, popsán v kapitole 6 a 7, který je rozdělen na teoretickou a praktickou část. Teoretická sekce prezentuje všeobecné informace o šifrovacím algoritmu RSA. V praktické části si uživatelé programu vyzkouší vlastní výpočetní úkony spojené s algoritmem RSA. Informace získané uživatelem v různých sekcích programu jsou dostačující k pochopení principu fungování tohoto algoritmu.

KLÍČOVÁ SLOVA

Šifrovací algoritmy, RSA, podpora výuky, faktorizace čísel, generování a testování prvočísel, digitální podpis

ABSTRACT

This thesis deals with new teaching software, which supports asymmetric encryption algorithms based on the issue of large numbers´ factorization. A model program was created. It allows to carry out operations related to encryption and decryption with an interactive control. There is a simple way to understand the principle of the RSA encryption method with its help.

The encryption of algorithms is generally analysed in chapters 1,2. Chapters 3 and 4 deals with RSA encryption algorithm in much more details, and it also describes the principles of the acquisition, management and usage of encryption keys. Chapters 5 suggest choosing of appropriate technologies for the creation of the final software product, which allow an appropriate way to present the characteristics of the extended RSA encryption algorithm.

The final software product is the java applet. Aplet is described in the chaprers 6 and 7. It is divided into a theoretical and practical part. The theoretical part presents general information about the RSA encryption algorithm. The practical part allows the users of the program to have a try at tasks connected with the RSA algorthm in their own computers. The last part of Java applet deals with the users´ work results. The information obtained by the user in the various sections of the program are satisfactory enough to understand the principle of this algorithm´s operations.

KEYWORDS

encryption algorithms, RSA, teaching support , number factorization , prime number generating and testing, digital signature

VYCHODIL, P. *Softwarová podpora výuky kryptosystémů založených na problému faktorizace velkých čísel*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. XY s. Vedoucí diplomové práce doc. Ing. Karel Burda, CSc.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma softwarová podpora výuky kryptosystémů, založeného na problému faktorizace velkých čísel, jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením tohoto projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce doc. Ing. Karlu Burdovi, CSc. za velmi cenné rady a odbornou pomoc při zpracování diplomové práce a její odbornou konzultaci.

V Brně dne ……. …….

Podpis autora

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ROVNIC

SEZNAM TABULEK

SEZNAM PŘÍLOH

1. ÚVOD

Kryptografie má velice dlouhou historii a rozsáhlé uplatnění ve světě kolem nás. Většina lidí potřebuje chránit svá data v počítačích nebo dát příjemci emailu najevo, že pisateli jsou právě oni. Nebo potřebují odeslat citlivé firemní informace nebo strategické plány. V soukromém životě zase potřebujeme utajit například informace o našem zdravotním stavu nebo platovém ohodnocení. Při všech těchto naprosto běžných činnostech využíváme právě kryptovacích technik. Již za dávných časů se používaly jednoduché šifrovací techniky, kdy se například dopisy s citlivými údaji psaly pomocí záměn písmen za jiné. Tyto metody jsou samozřejmě v dnešní době absolutně nedostačující.

V současnosti, kdy se velice dynamicky rozvíjí informační technika, získává stále více na důležitosti vývoj spolehlivých šifrovacích algoritmů, které jsou dostatečně rychlé a zároveň poskytují i odpovídající úroveň zabezpečení našich dat, ať se již jedná o data posílaná po síti, kde se klade velký důraz právě na rychlost šifrování a dešifrování nebo o data ukládaná na pevném disku našeho počítače. A zase naopak začínají dostávat do popředí potíže potencionálního prolomení algoritmů doposud považovaných za bezpečné.

Současné kryptovací technologie se dají rozdělit do dvou hlavních skupin a to na **symetrickou kryptografii** a **asymetrickou kryptografii.** Rozdíl mezi těmito dvěma skupinami je v použití klíčů pro šifrování a dešifrování. Do popředí pozornosti se samozřejmě dostávají i vědní obory zabývající se kryptoanalýzou – vědním oborem zabývajícím se bezpečností kryptovacích algoritmů a šifer.

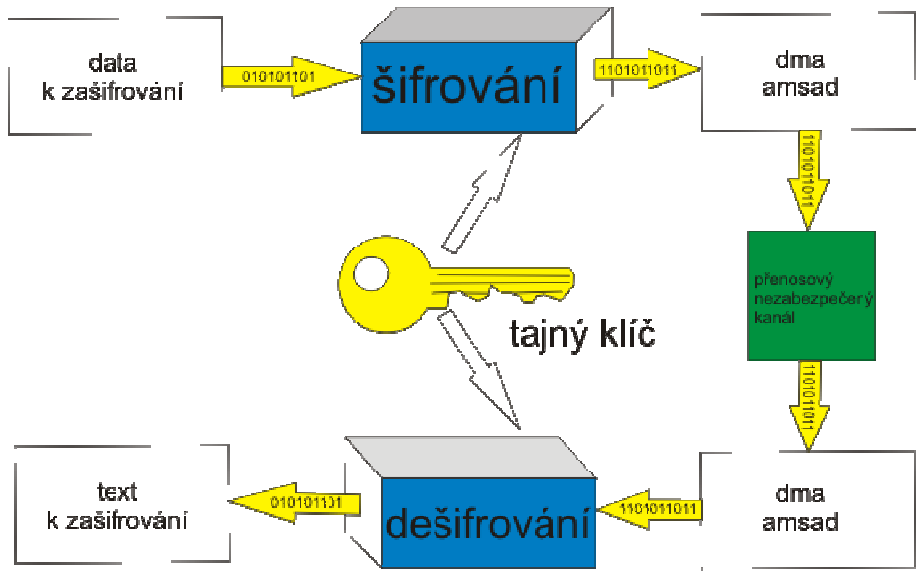
V mojí práci jsem se rozhodl zabývat hlavně právě **asymetrickými kryptovacími algoritmy,** protože tyto algoritmy jsou hojně rozšířené a chybí jejich vhodné komplexní zpracování, které by umožňovalo pochopit jejich princip jejich činnosti, výhody i případné nedostatky a samozřejmě také konkrétní aplikací do praxe.

2. KRYPTOGRAFICKÉ TECHNOLOGIE

Tyto kryptovací algoritmy můžeme rozdělit dle principu, na jakém šifrují a dešifrují data, na dvě hlavní skupiny. Zatímco *symetrické šifrovací algoritmy* používají jeden klíč k šifrování i dešifrování dat, *asymetrické šifrovací* algoritmy používají dvojici klíčů – *private key* a *public key.*

2.1 Symetrické kryptovací algoritmy

Princip těchto šifrovacích metod spočívá v tom, že do procesu krytování vstupuje dvojice klíče a šifrovaných dat. A při dešifrování se používají zašifrovaná data a opět stejný klíč jako při šifrování. Z toho jasně plyne *nutnost uchovat tento klíč v utajení*. S tímto jsou spojené problémy distribuce tohoto tajného klíče mezi strany, které data chtějí šifrovat. Princip synchronních kryptovacích metod je znázorněn na *Obr 1*.



Obr. 1: Synchronní šifrovací algoritmus

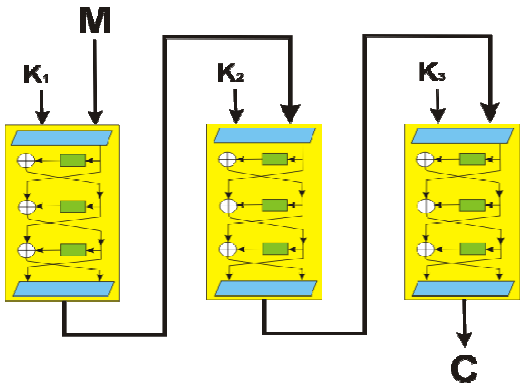
Dále bude uvedeno několik symetrických kryptografických metod, které jsou nebo byly hojně rozšířené, aby bylo možné pochopit lépe rozdíl mezi symetrickými a asymetrickými kryptografickými metodami.

2.1.1 DES

DES je z rodiny **blokových šifer** - Tyto blokové šifry je rodina kryptovacích algoritmů pracujících s daty v blocích. Další zástupci této skupiny jsou například: *AES,Blowfish,GOST,IDEA,RC2,RC5,Twofish,* která pracuje s daty v *64 bit* blocích pomocí šifrovacího klíče o délce *56 bitů*. Délka používaného klíče je vlastně rovněž 64bitů, ale každý 8. bit se používá pro paritní zabezpečení, a tudíž se při šifrování ignoruje a díky tomu vznikne 54-bitový klíč. DES byl navržen firmou *IBM*, ale jeho implementaci kontrolovala americká *NASA***.** V roce 1993 byl algoritmus DES prolomen na počítači v ceně 1 miliónu dolarů, který uměl realizovat prolomení DESu metodou totálních zkoušek v průměru za 3,5 hodiny. V současnosti není DES považován za bezpečný. Nyní se používají různé jiné deriváty DES algoritmu.

2.1.2 TripleDES

Triple DES je bloková šifra přímo vycházející z šifrovací metody DES. Jedná se vlastně o trojnásobnou aplikaci tohoto algoritmu. Viz.: Obr. 2. Používá délky klíče 168 bitů. Triple DES je oproti novějším algoritmům jako je například AES znatelně pomalejší. A v současné době je jeho používání na ústupu.



Obr. 2: Princip blokového šifrovacího protokolu TripleDES

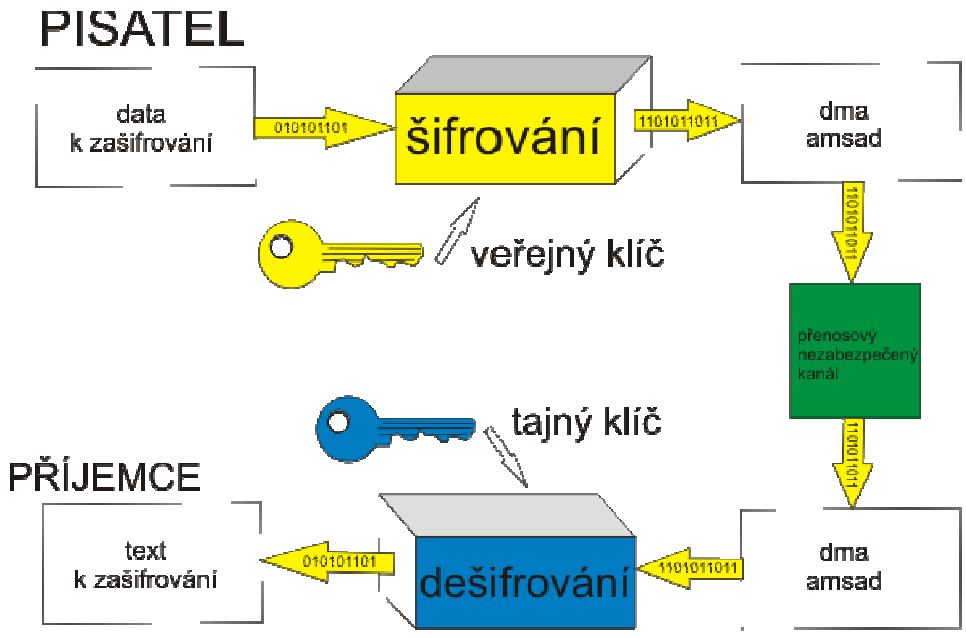
2.1.3 AES

Tento kryptovací protokol byl vyvinutý americkou vládou k šifrování dokumentů. Jeho velkou výhodou je vysoká rychlost šifrování. Do současné doby není veřejně znám žádný případ plného prolomení této šifrovací metody.

Tato šifra pracuje s délkou klíče 128, 192 nebo 256 bitů. Data se šifrují postupně, a to v pevné délce 128 bitů.

2.2 Asymetrické kryptovací algoritmy

Tyto algoritmy lze též nazývat jako kryptovací algoritmy s veřejným klíčem. Základní rozdíl mezi algoritmy symetrickými a asymetrickými je právě v politice klíčů. Zatímco symetrické algoritmy používají k šifrování i dešifrování jednoho tajného klíče. Asymetrické algoritmy mají dvojici klíčů: *veřejný* a *tajný viz.:* Obr. 3. Veřejný klíč má být co nejlépe přístupný a slouží k zašifrování dat. Naopak pomocí tajného klíče lze data zase dešifrovat. Nevýhodou ovšem je nutnost mít tajný klíč bezpečně uložen na serveru v místě, které není přes webové rozhraní přístupné. Toto ovšem je v případě užití pro šifrování dat, nikoli pokud se používají tyto kryptovací metody pro digitální podpis dokumentů nebo emailů.



Obr. 3: Asymetrické šifrovací metody - režim šifrování

Nevýhodou těchto asymetrických metod oproti symetrickým však zůstává, že jsou pomalejší, a to jak při šifrování, tak i dešifrování dat. Navíc je také nutné podotknout, že teoreticky lze z každého veřejného klíče výpočtem získat i klíč tajný. To ovšem se současnou výpočetní technikou není možné, pokud dodržíme určitou minimální délku klíče. Proto se zavedlo pravidlo, aby *tajný klíč nebyl získatelný v rozumném čase.* I když v současné době existují algoritmy pro zefektivnění kryptoanalýzy, například TWINKLE pro algoritmus RSA, těchto asynchronních kryptovacích algoritmů, stále by se jednalo o nereálné časy, které nejsou využitelné pro luštění, samozřejmě za předpokladu dodržení minimální délky klíče.

K tomuto lze využít matematických problémů, které mají obtížnost O závislou exponenciálně.

Jinak řečeno:

$$\boldsymbol{O}(\boldsymbol{c^n}), \boldsymbol{kde}\ \boldsymbol{c} > 1\ a\ n\ je\ parametr\ rozsahu\ \acute{u}\boldsymbol{lohy} \qquad \textbf{Rovnice 1.}$$

V současné praxi se k řešení toho problému nejčastěji využívá následujících matematických problémů:

- problém diskrétního algoritmu
- problém sčítání bodů eliptické křivky
- problém faktorizace velkých čísel

2.2.1 Problém diskrétního algoritmu

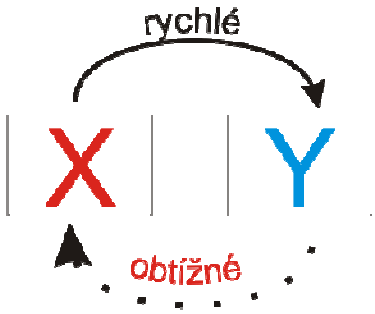
Problém diskrétního algoritmu, jinak zvaného taktéž jako Discrete Logarithm = DL, lze popsat pomocí následující matematické rovnice:

$$\boldsymbol{log}(\boldsymbol{n}\ \times \boldsymbol{mod}(\boldsymbol{p})) \qquad \textbf{Rovnice 2.}$$

Konkrétní rovnice má následující podobu:

Rovnice 3.

Kde, p je velkým prvočíslem a g je známé přirozené číslo . Pokud známe x je relativně snadné vypočítat y právě z rovnice 3. Ale právě při dešifrování je velice obtížné či prakticky nemožné vypočítat zpětně z **y** původní hodnotu **x**. To samozřejmě platí za předpokladu použití dostatečně velkého **p** *viz.: Obr4.* X znázorňuje zašifrovaná data, Y je kryptogram.



Obr. 4: Diskrétní logaritmy

Systémy diskrétních logaritmů se však nepoužívají pro šifrování dat, protože výsledný kryptogram je 2x delší než šifrovaná zpráva. Své uplatnění však našel pro digitální podpis a k ustanovení klíče, protože umožňuje domluvit klíč pro symetrické krytování při přenosu po nezabezpečeném přenosovém kanále. [1]

DiffieHellmanův algoritmus a příklad jeho užití je uveden v příloze A1.

2.2.2 Problém Eliptických křivek

Výzkum algebraické geometrie a teorie čísel eliptických křivek je znám více než 150 let, ale až v roce 1985 zaměstnanec *firmy IBM* a nezávisle na něm i na *University of Washington*, konkrétně Victor Miller z IBM a Neal Kobliz z University of Washington, na možnost jejího použití v systémech asymetrického šifrování.

V té době se ale na metodu eliptických křivek pohlíželo jako na neperspektivní z důvodu již vyřešeného problému diskrétního algoritmu. Časem se však přišlo na těžkopádnost systému typu RSA. Tehdy se začalo uvažovat o nasazení eliptických křivek, které mají v této oblasti výrazně lepší parametry. Tato rychlost je velmi důležitá například u *linkových šifrovacích protokolů*. A zase naopak jejich jednoduchost výpočtu a paměťová nenáročnost je předurčuje k použití u čipových karet. [1]

Hlavní výhody metody eliptických křivek jsou tedy následující:

- vysoká míra zabezpečení, mají nízké nároky na výpočty, šířku pásma i paměť.
- rychlost šifrování i podepisování, a to jak u hardwarové, tak i u softwarové realizace.

2.2.3 Problém faktorizace velkých čísel

Jedná se o využití algoritmu RSA, jeho podrobnému prozkoumání budou věnovány následující kapitoly.

3. RSA

Je založeno na jednoduché myšlence, že je snadné vynásobit dvě dlouhá prvočísla, ale bez jejich konkrétní znalosti není možné zpětně provést rozklad tohoto výsledného čísla na původní prvočísla, ze kterých bylo získané. Součin těchto čísel je nazýván jako veřejný klíč – *public key*. Tato obě prvočísla ale potřebujeme při dešifrování dat.

Algoritmus RSA je rovněž hojně používán k digitálním podpisům. Patří tedy do rodiny asynchronních kryptovacích algoritmů, které při šifrování a dešifrování nebo podpisu elektronických dokumentů používají dvojici klíčů: veřejného a privátního.

3.1 Historie RSA

RSA - jsou počáteční písmena autorů této metody: Ron Rivest, Adi Shamir a Joe Adleman, bylo objeveno roku 1977. Algoritmus RSA byl až do roku 2000 patentován na území USA. RSA bylo vyvinuto na MIT - Messachussets Institute of Technology. V současné době byla prolomena délka klíče o délce 640 bitů (193 dekadických číslic). Firma RSA zveřejňuje výzvu k prolomení algoritmu RSA. V případě prolomení šifry s délkou klíče 704 bitů se jedná o honorář 30 000 dolarů, dále jsou potom v současnosti vypsány honoráře na délku klíče 768 bitů (50 000 dolarů) a 1024 bitů (100 000 dolarů).

3.2 Výhody a nevýhody RSA

Synchronní algoritmy mají jednu zásadní nevýhodu, tou je nutnost distribuce a domluva na použití konkrétního tajného klíče, který se v následné komunikaci bude používat k šifrování a dešifrování. Odstranění tohoto problému je velmi silnou vlastností algoritmů RSA. Další výhodou je, že při komunikaci například se serverem s tisícem uživatelů, stačí mít na serveru jeden veřejný klíč a druhý tajný. Tudíž je možné veškerou komunikaci mezi uživateli bez problému šifrovat. Kdokoli tedy chce poslat tajnou zprávu serveru, použije jeho veřejný klíč, který je přístupný například přes web rozhrání serveru. Jednou z dalších možností aplikací je možné použití pro tzv. digitální podpis, což je hash zprávy zašifrovaný tajným klíčem. Využívá se při tom takzvaných hash funkcí. Ty udělají otisk (hash) zprávy. Princip hashe je takový, že i když na vstup přivedeme libovolně dlouhou zprávu, dostaneme její otisk dlouhý 128 nebo 160 bitů.

Pokud bychom ve zprávě změnili nějaké údaje, byť jediný znak, hash funkce nám na svém výstupu vytvoří úplně jiný otisk. Stejně tak je nemožné, nebo lépe řečeno vysoce nepravděpodobné, vytvořit zprávu, která má stejný hash. Bylo by to výpočetně stejně náročné jako zprávu rozluštit bez znalosti klíče. Tento výsledný hash zprávy se zašifruje tajným klíčem. Tomuto zašifrovanému otisku se říká digitální podpis. Nakonec se digitální podpis připojí ke konci původní podepisované zprávy.

Bohužel asymetrická kryptografie, přes výhodu své politiky dvou klíčů, má rovněž i své nevýhody. Je značně (až 1000x) pomalejší než kryptografie symetrická. O kolik je pomalejší, je nejvíce ovlivněno délkou použitého klíče. [1]

Například pro AMD Opteron 1,6 GHz byly naměřeny následující rychlosti:

Režim	**Délka klíče**	**Potřebný čas**	**rychlost**	**poznámka**
[-]	**[bit]**	**[ms]**	**[kbps]**	**[-]**
RSA	1024	4,77	215	-
RSA	2048	28,4	72	2x delší a 3x pomalejší
AES	128	262×10^{-6}	488×10^{3}	2270/6780x rychlejší

Tabulka 1: Porovnání rychlostí RSA

Další nevýhodou je, že komunikující strana musí mít jistotu, že veřejný klíč, kterým se chystá svá data zašifrovat, doopravdy náleží serveru, se kterým opravdu komunikovat chce, a tudíž, že se nejedná o podvrh.

3.3 Úloha klíčů v RSA

Generování klíčů je pro symetrické algoritmy (AES, DES) jednoduchou úlohou. Klíčem může být prakticky cokoli, co je dostatečně dlouhou náhodnou hodnotou. Proto se také často používají jednorázové klíče. Avšak generování klíčů pro RSA je dosti náročný výpočet, protože se jedná o extrémně náročné výpočty s velkými prvočísly. Tyto klíče se proto uchovávají a dochází k jejich opakovanému použití.

V případě RSA algoritmu si při generování klíče musíme také pečlivě zvolit délku klíče. S jeho vzrůstající délkou stoupá náročnost prolomení šifry, ale bohužel také i náročnost všech prováděných operací - generování klíčů, šifrování i dešifrování. A to bohužel ne lineárně ale:

Zvýšíte-li délku klíče na dvojnásobek, zvýší se v průměru:

- doba generování klíče 16×
- doba potřebná pro dešifrování (a další operace s tajným klíčem) 8×
- doba potřebná pro šifrování (a další operace s veřejným klíčem) 4×

Vhodnou délku klíče tedy musíme opravdu pečlivě volit. Správná délka závisí na mnoha faktorech. Jednou ze zásadních je například doba, po kterou mají být naše data chráněna. Je sice samozřejmě možné, data rozšifrovat a zase zpětně zašifrovat delším klíčem, to je ale značně časově náročně, pokud by se jednalo o velké množství dokumentů. Je tudíž vhodné počítat s vývojem výpočetní techniky v dnešní době. Pokud aplikace používá specializovaný hardware, je možné si dovolit použití delšího klíče, protože operace spojené s šifrováním a dešifrováním budou probíhat daleko rychleji.

Klíč délky 512 bitů byl prolomen v roce 1997 na hardwarovém zařízení v ceně jednoho miliónu dolarů, a to již v době okolo osmi měsíců. Nyní se doporučuje použití minimální délky klíčů 1024 bitu. Pro důležité informace potom 2048 bitů. Nejdelší klíč, který můžeme efektivně používat na běžném hardwarovém vybavení, má délku 4069 bitů.

3.3.1 Distribuce a uchovávání klíčů RSA

Generování klíčů

Aby bylo možné vygenerovat klíče pro RSA šifrovanou komunikaci, postupuje se následujícím způsobem:

Zvolí se dvě náhodná čísla správné velikosti. A následně se otestují pomocí testu prvočíselnosti. Test prvočíselnosti spočívá v ověření, zda li dané číslo opravdu prvočíslem je, či nikoli. Vzorová funkce k ověření prvočíselnosti čísla je v příloze A2.

Vzhledem k možnému potencionálnímu útoku Fermatovou faktorizací – Viz.: příloha A1 – Fermatova faktorizace, se musí zvolit dostatečně rozdílná čísla *p* a *q*. Další potencionální hrozbou by bylo, pokud se *p-1* nebo *q-1* rozkládají na malá prvočísla. Potom by mohly být fakticky velice rychle faktorizovány, tudíž by tyto hodnoty *p* a *q* rozhodně neměly být používány.

Důležité také je, že by se neměla používat, jako vyhledávací metoda taková, která umožní získat nějaké informace vedoucí k prolomení šifry. Generování začíná náhodným vygenerováním čísel. Tato čísla musí být zároveň i nepředvídatelná. Náhodnost a nepředvídatelnost není shodným požadavkem. Protože spousta náhodných veličin je popsána určitými matematickými modely.

Stejně tak je dobré využít náhodného generátoru, jehož přesný algoritmus není nikde zveřejněn, protože případná publikace umožňuje získat potencionálním útočníkům informace, které by mohli použít a díky nimž například jsou schopni zúžit výběr náhodných čísel. K těmto informacím by jinak, v případě utajeného generátoru, neměli přístup.

Rovněž důležitým požadavkem je, aby tajný klíč měl dostatečnou délku. Wiener v roce 1990 dokázal, že pokud platí rovnice 4. Pak *d* může být účinně vypočítané z *n* a *e*.

$$\boldsymbol{q} < p < 2 \;\; a \; souč\boldsymbol{asn}ě \; \boldsymbol{d} < \frac{\sqrt[4]{\boldsymbol{n}}}{\boldsymbol{3}} \qquad \textbf{Rovnice 4.}$$

Přitom do současnosti není znám postup útoku proti malému veřejnému exponentu - v případě použití správné výplně. Pokud ale výplň nepoužijeme žádnou nebo nevhodnou, potom použití tohoto veřejného exponentu nese vyšší riziko útoku. NIST návrh FIPS PUB 186-3, toto doporučení bylo vydáno v roce 1990, nedovolí používat veřejné exponenty menší než e = 65 537, ale tento požadavek žádným způsobem vědecky nepodkládá.

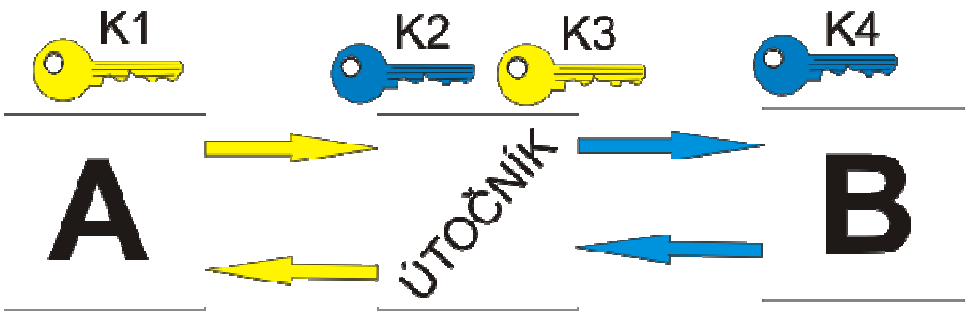
Postup krok po kroku je následující:

1. Volba dvou různých velkých náhodných prvočísel p a q.
2. Spočítá jejich součin
3. Spočítá hodnotu Eulerovi funkce $\varphi(n) = (p - 1)(q - 1)$.
4. Zvolí celé číslo e menší než $\varphi(n)$, které je s $\varphi(n)$ nesoudělné.
5. Nalezne číslo d tak, aby platilo $de \equiv 1 \pmod{\varphi(n)}$.
6. Jestli e je prvočíslo tak $d = (1+r*\varphi(n))/e$, kde r = [(e-1)$\varphi(n)$^(e-2)]

Veřejný klíč je dvojice (e, n), přičemž n se označuje jako *modul*, e jako *šifrovací* či *veřejný exponent*. Soukromý klíč je dvojice (d, n), kde d se označuje jako *dešifrovací* či *soukromý exponent*. Prakticky se používá k uchování mírně odlišná forma, která umožňuje rychlé zpracování.

Distribuce klíčů

Při distribuci klíčů je nutné mít na paměti jednu velice zásadní věc. Je bezpodmínečné zařídit to, aby nenastal útok typu man in *the middle attack*! Pokud to není možné, následná identifikace útočníka je prakticky nemožná. A to bohužel pro obě dvě strany.



Obr. 5: RSA princip podvržení klíče

V případě, že útočník dokáže podvrhnout klíč neboli vydávat svůj veřejný klíč za klíč jiného účastníka komunikace, jedná se o kritické selhání šifrování. Celý útok probíhá následujícím způsobem. Účastník **A** má v plánu komunikovat s účastníkem **B**.

Pokud ovšem útočník zachytí zasílaný veřejný klíč K2, který patří účastníkovi A, avšak který je určen pro účastníka B a dále pak místo klíče K2 zašle svůj podvržený veřejný klíč K4 účastníkovi B - ani jedna ze stran není schopná zachytit nějakou změnu. Při následném zaslání zprávy je nejdřív zašifrována pomocí tajného klíče K1 účastníka A, následně ji však odchytí a dešifruje ÚTOČNÍK pomocí veřejného klíče K2 účastníka A. Se zprávou je schopen libovolně nakládat, následně ji zašifrovat svým podvrženým tajným klíčem K3 a zaslat účastníkovi B. V případě odesílání zprávy od B k A se jedná o stejný princip. Na obrázku 5 je komunikace šifrovaná pomocí pravého klíče serveru znázorněná žlutou šipkou a pomocí podvržených klíčů modrou. Ochrana proti takovému typu útoku spočívá v použití digitálního podpisu a certifikátů. [1][2]

3.3.2 Uchovávání a ochrana soukromých klíčů

Typicky se RSA klíče uchovávají v takzvaných kontejnerech. Tyto kontejnery jsou vlastně obyčejné soubory uložené na harddisku počítače. Tyto soubory jsou chráněny systémovými prostředky operačního systému. Existují ovšem i specializované hardwarové součástky, například šifrovací token nebo čipové karty, které se do počítače připojují přes interface, například bluetooth nebo typičtěji USB.

Při ochraně tajemství je nutné mít na paměti jedno ze základních pravidel kryptografie. **Zašifrováním tajné informace tato informace nezmizí, pouze zmenšíme jeho objem.** Místo zajišťování řádově GB nebo dokonce TB dat, stačí chránit třeba jediný tajný klíč. Ochrana tohoto privátního klíče je nejslabším článkem jakýchkoli principů kryptografických a šifrovacích metod všeobecně. Největším nebezpečím ztráty tajné informace je právě kompromitací tohoto tajného klíče. [1][2]

Nejbezpečnější metodou je samozřejmě to, pokud jsme schopni fyzicky privátní klíč odstranit z informačního systému, pokud se nepoužívá. Toto nám umožní například použití čipových karet nebo šifrovacích tokenů. Tyto totiž obsahují i kryptografický procesor a všechny operace se provádí v rámci této karty, tokenu. Tajný klíč tudíž nikdy neopustí čipovou kartu, je vygenerován při její výrobě a pokud se útočník pokusí tajný klíč z karty (*tokenu)* získat, jsou tyto karty vyrobeny tak, že dojde ke zničení karty i privátního klíče. [2]

RSA – typicky všechny asymetrické kryptovací algoritmy nabízejí další možnost zabezpečení. Protože není většinou nutné, aby počítač, který data šifruje, nutně musel mít možnost i tato data rozšifrovat. Například typicky údaje udávané přes internet prostřednictvím formulářů. Data se zašifrují pomocí veřejného klíče a uloží na server. Tajný klíč používaný k dešifrování je uložen v místě, které není přes web server přístupné, nebo rovnou lépe na jiném PC nepřipojeném do sítě. A data, pokud je nutné je dešifrovat se na tento PC přenesou například pomocí média *(DVD,CD,FLASH).*

3.4 RSA v režimu pro šifrování a dešifrování

Jeden ze základních možných způsobů využití algoritmu RSA je šifrování a dešifrování dat.

3.4.1 Zašifrování zprávy

Účastník A musí zaslat tajnou zprávu Z. Tato zpráva se převede nějakým předem určeným postupem na číslo m. Musí být splněna základní podmínka:

$$\mathbf{m < n.}$$

Šifrovaným textem odpovídajícímu Z je potom číslo, které získáme z následující rovnice:

$$\boldsymbol{c = m^e \; mod \; n} \qquad \textbf{Rovnice 5.}$$

Tahle šifrovaná data se zašlou prostřednictvím nezabezpečeného kanálu. Pokud je možné vyloučit útok typu man in the middle, zůstanou naše data utajená.

3.4.2 Dešifrování zprávy

Účastníkovi komunikace B dojde zašifrovaný text od A, tento text bude označen jako kryptogram c. Původní zprávu je schopen získat z rovnice 6. [1][2]

$$\boldsymbol{m = c^d \bmod n} \qquad \textbf{Rovnice 6.}$$

To že je získána původní zpráva je podloženo následující rovnicí:

$$c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}. \qquad \textbf{Rovnice 7.}$$

Protože:

$$ed \equiv 1 \pmod{p-1} \text{ a } ed \equiv 1 \pmod{q-1},$$

Z Fermatovy věty platí:

$$m^{ed} \equiv m \pmod{p}$$

a zároveň

$$m^{ed} \equiv m \pmod{q}$$

Jelikož p a q jsou různá prvočísla, díky *čínské větě o zbytcích* je dáno

$$m^{ed} \equiv m \pmod{pq}.$$

Tudíž:

$$c^d \equiv m \bmod n. \qquad \textbf{Rovnice 8.}$$

3.4.3 Příklad výpočtu

Z důvodu zjednodušení číselných výpočtů, jsou použita nízká čísla, v praxi jsou použita čísla o několik řádů (desítek) vyšší.

Zadání: $p = 61$ (první prvočíslo) $q = 53$ (druhé prvočíslo) $n = pq = 3233$ (modul, veřejný) $e = 17$ (veřejný, šifrovací exponent) $d = 2753$ (soukromý, dešifrovací exponent) Pro zašifrování zprávy 123 probíhá výpočet:

$$\text{crypt}(123) = 123^{17} \bmod 3233 = 855$$

Pro dešifrování pak:

$$\text{decrypt}(855) = 855^{2753} \bmod 3233 = 123$$

3.5 RSA v režimu digitálního podpisu

3.5.1 Co to je elektronický podpis

Definice dle legislativy české republiky zní:

Elektronickým podpisem se rozumí údaje v elektronické podobě, které jsou připojené k datové zprávě nebo jsou s ní logicky spojené, a které umožňují ověření totožnosti podepsané osoby ve vztahu k datové zprávě. Jinak řečeno umožňuje jednoznačně identifikovat identitu odesílatele zprávy. Zákon i práce rozlišují dva typy digitálního podpisu. [2][3]

- elektronický podpis – podpis dohodnutý subjekty na základě smluvní formy
- zaručený elektronický podpis – není třeba smluvní dohody, ale tento podpis je potvrzen takzvanou certifikační autoritou (třetí stranou), která ověřuje a garantuje autentičnost tohoto elektronického podpisu

Využití digitálního podpisu je více než zřejmé, přesto zde budou uvedeny hlavní možnosti použití tohoto elektronického podpisu:

- jednání s úřady
- bezpečné obchodování na internetu
- zabezpečení emailové korespondence

Dostáváme se tedy k dalšímu široce využívanému režimu algoritmu RSA. Tímto způsobem využití je digitální podpis. Jedná se vlastně o zrcadlové použití šifry RSA. Jak takový digitální podpis vypadá je zřejmé z Obr. 6.

3.5.2 Princip digitálního podpisu

Pokud je nutné podepsat nějakou zprávu digitálním podpisem, nejdříve si účastník komunikace A spočítá hash zprávy, tento hash zprávy zakóduje svým tajným klíčem. Tomuto se říká digitální podpis. Podpis se připojí k původní podepisované zprávě. Účastník B si po přijetí podepsané zprávy spočítá rovněž její hash a dešifruje si hash obdržený ve zprávě pomocí veřejného klíče účastníka A. Následně tyto dva údaje porovná. V případě, že jsou stejné, je zaručeno, že zprávu zasílá účastník A.

Důležitou vlastností digitálního podpisu je to, že účastník A nemůže popřít skutečnost, že zprávu podepsal doopravdy on. K šifrování hashe místo celého obsahu zprávy se přistupuje, protože kdybychom zašifrovali zprávu celou a tu pak připojili k podepisované, docházelo by k neúměrnému nárůstu dat. Takto se výsledná zpráva zvětší pouze o pár bytů. [1][2][3]

3.6 Bezpečnost RSA

Další důležitou vlastností RSA je bezpečnost této kryptovací metody. Ta je založena na dvou matematických problémech.

1) faktorizace velmi vysokých čísel
2) RSA problém

Rozšifrování zprávy zašifrované pomocí algoritmu RSA je v současnosti v podstatě neproveditelné, protože není doposud znám algoritmus, podle kterého by šlo prolomení šifry provést. Aby nešla tajná zpráva dešifrovat ani částečně, je nutné použít dobré schéma vyplňování. [1][2]

RSA problém je úloha získání kořenu modulu množiny *n* obnovením původní hodnoty *m* jako $m^e = c \bmod n$, kde (*e,n*) jsou RSA veřejného klíče a *c* je kryptogram získaný prostřednictvím RSA. V současné době je považována jako nejlepší metoda k získání kořenu takzvaný faktor modulu *n*. V případě, že je možné obnovit tyto primární faktory, je možné, aby útočník vypočítal i tajný exponent *d* z veřejného klíče. (*e,n)*. Poté mu nic nebrání přistoupit k dešifrování *c* pomocí standardního postupu dešifrování při znalosti tajného klíče. Útočník je totiž schopen pomocí faktoru *n* do *p* a díky tomu spočítat i rovnici (p-1) (q-1). To mu umožní bezpečně stanovit *d* pro *e*. Nebyla však doposud nalezena metoda s polynomiální časovou složitostí, která by toto dokázala. Nutno však poznamenat, že rovněž nebylo dokázáno, že tato metoda neexistuje. [1][2]

Nejdelším úspěšně prolomeným číslem do roku 2005 bylo číslo o délce 663 bitů, a to bylo prolomeno použitím distribučních metod. Vzhledem k tomu, že běžně používané RSA klíče mají délku 1024 – 2048 bitů, je možné považovat tuto kryptovací metodu za bezpečnou, pokud je použit dostatečně dlouhý klíč, i když někteří experti věří, že délka klíče 1024 bitů bude v blízké době kompromitována. Každopádně však není známa žádná metoda, která by mohla vést k prolomení klíče o délce 4096 bitů. Pokud by však byl použit klíč o délce 256 bitů nebo kratší, bylo by možné ho faktorizovat na obyčejném počítači v průběhu několika hodin a s použitím běžného softwaru. Jestliže je n = 512 bitů, je nutné k faktorizaci použít několik (řádově stovky) počítačů. [1][2]

Teoreticky navrhnuté zařízení TWIRL, které popsali Shamir a Tromer v r. 2003 rozpoutalo bouřlivou diskuzi o bezpečnosti klíče o délce 1024 bitů. Proto se nyní doporučuje použití klíče o minimální délce 2048 bitů. [1]

V roce 1993 v publikaci Shorův algoritmus, je vysloven předpoklad, že kvantový počítač by mohl v principu vykonávat faktorizaci v polynomiálním čase, což by algoritmus RSA učinilo nepoužitelným. Avšak realizace kvantového počítače je vzdálenou budoucností a v současnosti se o bezpečnost šifrování pomocí RSA není třeba obávat. [2]

3.7 Použití RSA

PGP – pretty good privacy, využívá rovněž asymetrické kryptografie. Je to nástroj sloužící k šifrování podepisování dešifrování nebo ověření podpisu této zprávy. PGP používá další pár klíčů, jeden tajný pro podepisování a dešifrování vlastních zpráv a druhý veřejný pro příjemce k ověření podpisu. Dále poté potřebuje veřejné klíče příjemců. Tyto používá k ověření identity ostatních klientů a k šifrování zpráv zasílaných klientům. Datové úložiště, kde se tyto zmiňované klíče uchovávají, se nazývá PGP KEYRING. V PGP jsou používány dva takové soubory. Soubor sloužící k uchování veřejných klíčů a druhý k uchování tajného klíče (klíčů). PGP totiž může disponovat větším množstvím klíčových párů. PGP totiž umožňuje jakoby několik úrovní zabezpečení. Jednu dvojici klíčů je možné použít například při komunikaci s přáteli a další při citlivější při komunikaci *například firemní*, které mají větší délku.

PGP používá, rovněž k ochraně tajného klíče heslo, takzvanou phassphrase. Toto heslo by mělo mít v ideálním případě několik slov. Toto heslo slouží k ochraně před zneužitím tohoto tajného klíče v případě například odcizení z počítače. Toto heslo se tudíž musí zadávat při šifrování nebo dešifrování dat.

Ukázka digitálního podpisu vygenerovaného programem PGP pro nekomerční použití:

```
ahoj
Růža
-----BEGIN PGP MESSAGE-----
Version: PGPfreeware 7.0.3 for non-commercial use <http://www.pgp.com>

qANQR1DBwU4D0AvtMHA1qbAQCACLbsQtdFx0yA3dKAu9PAcZ95OEiUeG2qTIUroo
VUKLbvPmhhmgpGAEBKdn/gdQLQRGedUpdUvobLCufBgfpuI0TruRg3TA96yaBT9+
BjSaTy/AikZvJOyY6vtEniQ7A3UmL2B+anFf7p5PXFzMcglavmZtUWldOcC87I9V
mJxBVQZySFFFZxuV17U33NuJMeC2Nwc32VB+1RJdvHSiWVbnsfh40xOVNTzkk5Fl
agYwx3hGwgwbItekV3g9E/8TV9nUonCYIml8nlnVKwg23RmGxiK9Wok4yhoMoKAd
r9bDAKuN91YmhvbL5wqmiWv+haMF7d0dAwRa/P8G3u9l50xiB/9uz/789V7ryl5F
YYGk9UjtJR6zxxJ6kyaQZj//s+1vFEEh/JPVujMWVxYvft8T/75KWlPei4oJQRwO
24L5w7ZCFgH46VhEq93i9gRgwBp1ugYPO4/+T+ynyFnDWz6Fs5oGtifEFUF5Xi7z
PCu1Rbw2AJTVcb5/yyWHwYRJF6Zm0EJ3O+TROi5qSuswSC68Ju5ZMVB0GPI4omNV
yMHsZUhASr6i3rXUw/vypi4cQ/gn0AsyODzK1PsMNfGxe/fX8XxiUmRkm+C6nvRM
8LLrhNCFdvvVHMI+SoOhbLomBL0WunFOMOSQEfDW5h5B9LcIqO2Qc7K3HoxsClu8
M7j5IqG0ySNj5A1KlTgEIosnC6uLpK6/IHkEcdY5UPEvFu8soaABb7UA/Q==
=916t
-----END PGP MESSAGE-----
```

Obr. 6: Ukázka digitálního podpisu vygenerovaného programem PGP

4. GENEROVÁNÍ, TESTOVÁNÍ VELKÝCH PRVOČÍSEL A MOCNIN

Doposud je známé velké množství algoritmů, pomocí kterých lze prvočísla generovat nebo testovat. Bohužel, velká většina z nich je příliš paměťově nebo naopak časově náročná, než aby byly tyto algoritmy využitelné pro použití v šifrovacích metodách.

Definice prvočísla:

Prvočísla jsou čísla, která jsou různá od nuly. A kromě jedničky a sama sebe nemají žádného celočíselného dělitele. Pokud je potřeba otestovat, zdali dané číslo je opravdu prvočíslem, je dokázáno, že stačí testovat jeho dělitele do velikosti druhé odmocniny z daného čísla. [2]

4.1 Eratosthenovo síto

Jedním z možností generování prvočísel je použít algoritmu tzv. **Eratosthenova síta**. Tento algoritmus funguje tak, že na počátku jsou všechna čísla z požadovaného rozsahu označena jako prvočísla. A tato se potom postupně prosévají sítem, dokud opravdu nezůstanou jen prvočísla. Postupně se berou čísla z nejnižších pozic a testuje se, jestli se jedná či nejedná o prvočíslo. V případě, že se opravdu o prvočíslo jedná, všechny jeho násobky z daného rozsahu se označí jako ne-prvočísla. Realizaci tohoto algoritmu naleznete v příloze A2.

4.2 Alternativy

Další možností je napsat si svoji funkci pro generování prvočísel. Jednou z možných realizací je například implementace v jazyce C:

```
#include <stdio.h>
#include <stdlib.h>

void main(int argc, char *argv[])
{
   if(argc!=3)
   {
    printf("usage: RH5 from to");
    return;
   }

   int i, j, k,from(atoi(argv[1])), to(atoi(argv[2]));

   printf("Prime numbers between %d and %d",from, to);
```

```
   for(i=from;i<=to;i++)
   {
    for(j=2, k=i ;j<k;j++, k=i/j)
      if(0==(i%j))break;
      if(i==1 || j>=k)printf("\n%d is a prime number",i);
   }
}
```

4.3 GCD – Greatest Common Divisor

V češtině je také nazýván jako největší společný dělitel. Jedním ze základních požadavků na vygenerovaná prvočísla je, že by měla být blízko od sebe. A neměla by se o mnoho lišit. Jedním ze způsobů, jak tohle zajistit, je použití Euklidova algoritmu. Ideální je, vzhledem k vygenerovanému poli prvočísel, aby nám tento Euklidův algoritmus našel dvě čísla z tohoto pole. Viz následující příklad: [1]

```
 /* navraci GCD z x,y */
 int gcd(int x, int y)
{
  int g;
   If(x<0)
    x=-x;
   if(y<0)
    y = -y;
if(x+y == 0)
 ERROR;
g=y;
while(x>0) {
  g=x;
  x = y % x;
  y = g;

 }
 return g;
}

 Int mult_gcd(int m, int x)  {
   size_t I;
   int g;

  if(m<1)   {
   return 0;
   g = x[0];
   for(i=1;i<m;++i)
   g = gcd(g,x[i]);
   /* optimalizace co usetri az 60% casu */
   if(g ==1)
   return 1;
  }
return g; }
```

4.4 Počítání modulo velkých čísel a jejich mocnin

Vzhledem k vlastnostem mocniny je jasné, že umocněním velkého čísla získáváme ještě mnohem větší číslo. V případě, že vyjdeme z rovnice 9, tak platí že:

Rovnice 9.

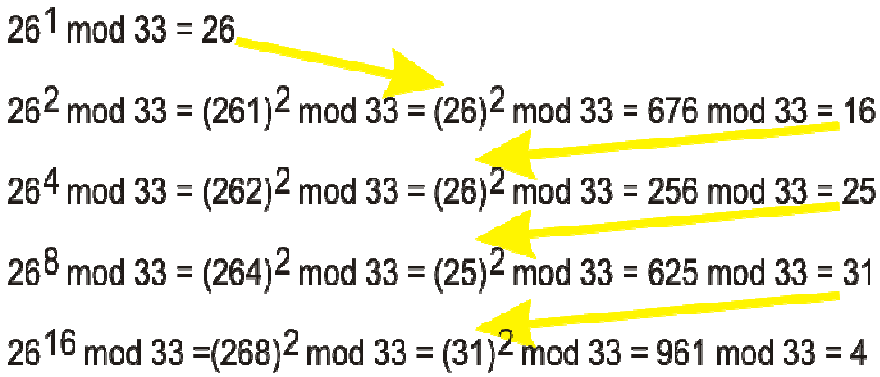
Poté je možné psát:

mod n

Z předcházející rovnice plyne dále:

Rovnice 10.

Rovnice 10 se používá při výpočtu velkých mocnin. Nutno dodat, že toto zjednodušené odvození vychází z čínské věty o zbytcích. Konkrétní příklad výpočtu je znázorněn na Obr. 7. [1][2]



Obr. 7: Princip výpočtu velkých mocnin

5. IMPLEMENTACE RSA PRO PODPORU VÝUKY

V případě, že má aplikace sloužit hlavně pro podporu výuky, musí z ní být hlavně patrny názorně používané principy. Musí být jasně zřejmé, ve kterém kroku se získá například kryptogram zprávy, ve kterém máme k dispozici tajný klíč nebo naopak již zašifrovaná data. Zároveň by aplikace měla využívat moderních metod, v tomto případě programovacích jazyků, aby studenti dostali do rukou potencionální a rozvíjející se techniku a aby čas, který věnovali studiu tohoto problému, nebyl jen pouhým mrháním, ale vedl k možnému využití v pozdější praxi. [4]

Algoritmus RSA je více či méně podporován ve všech současných programovacích jazycích. Z důvodu všeobecné podpory výuky a hlavně na základě stanovených parametrů a požadavků na program, byl jejich výběr zredukován na volbu mezi nyní již klasickým jazykem C (C#, C++) a Javou. Nakonec vyhrál jazyk Java, a to z níže uvedených důvodů. [5]

5.1 Pro implementaci zvolen jazyk Java

Rozhodl jsem se také pro implementaci v programovacím jazyku Java, protože tento jazyk se v současné době začíná čím dál tím více používat. Nachází široké uplatnění právě v síťových aplikacích a internetu. Pro jeho volbu rovněž hovoří i další parametry jako jsou například: [5]

- platformová nezávislost
- vývojové prostředí plně zdarma
- díky optimalizacím kódu jsou programy napsané v Javě srovnatelně rychlé jako v jazycích typu C nebo C++
- v současnosti rozšířené a perspektivně se rozvíjející.
- dobrá podpora kryptografických metod

Java má ovšem i své nevýhody:

- delší doba startu aplikace a paměťové nároky prostředí
- vykreslování grafických aplikací. SWING je totiž oproti nativním vykreslovacím toolkitům pomalejší.

Po zvážení všech výhod a nevýhod padla volba na tento programovací jazyk. Jedním z nejdůležitějších rozhodujících faktorů bylo to, že Java se začíná široce na školách vyučovat. Tudíž je předpoklad, že studenti nebudou, nebo lépe řečeno neměli by mít, potíže s pochopením zapsaných algoritmů, ale budou se moci věnovat přímo, bez odvádění pozornosti, pochopení daného problému. Dalším z faktorů proč použít Javu byla možnost bezproblémového vytvoření výukového Java apletu a jeho následného umístění na internet, kde bude bez problémů dostupný ke studiu z domova nebo kolejí. Rovněž podstatným prvkem je výborná podpora jazyka pro kryptosytém RSA. [5]

5.2 Vývojová prostředí pro Javu

K psaní programů v jazyce Java je k dispozici široká škála nejrůznějších programů. Tyto produkty mají různou pořizovací cenu. Je ovšem nutné dodat, že mezi cenou a kvalitou vývojového prostředí není vždy přímá úměrnost. Úspěšné programy v Javě lze tvořit i v obyčejném poznámkovém bloku. Ten ovšem neumožňuje ani zobrazení syntaxe programu. A při jeho použití se jedná víceméně pouze o nouzové řešení. Již vylepšenou verzí poznámkového bloku se jedná o zdarma dostupný program PsPad.

Tento program již podporuje širokou řadu programovacích jazyků a je to jeden z nejoblíbenějších vývojových prostředí programátorů PHP. Umožňuje zvýraznění syntaxe a v omezené míře i našeptávání funkcí. Stále však není možné mluvit o vhodném vývojovém prostředí k tvorbě plnohodnotných projektů v Javě. Protože neumožňuje přímou kompilaci zdrojových kódů. Rovněž je v něm obtížné vytvoření designu aplikace.

Dalším z již plnohodnotných dostupných vývojových programů primárně určených pro Javu je BlueJ. Jedná se o integrované prostředí určené primárně pro výuku objektově orientovaného programování v Javě. Nebo jEdit což je programátorský editor napsaný v Javě s podporou editace velkého množství programovacích jazyků a formátů. Po instalaci je vhodné do jEdit doinstalovat

paginy. Při psaní v jazyku Java jsou konkrétně vhodné tyto pluginy: JCompiler (volání překladače přímo z editoru), JavaStyle (formátování zdrojového kódu), Console (volání programů z editoru), PMD (kontrola konvencí), AntFarm (podpora projektů přes ant), CodeAid (doplňování kódu). [5]

Mezi dva nejdůležitější, nejlépe propracované produkty, nabízející širokou množinu pro programátora užitečných funkcí s možností přímého grafického návrhu a v neposlední řadě vynikající technickou dokumentací se řadí NetBeans a Eclipse. Právě z těchto dvou bylo vybíráno prostředí pro tvorbu konečného programu této diplomové práce. Volba padla na produkt NetBeans z důvodu toho, že subjektivně hodnoceno, NetBeans je přehlednějším a uživatelský příjemnějším prostředím než Eclipse. [5]

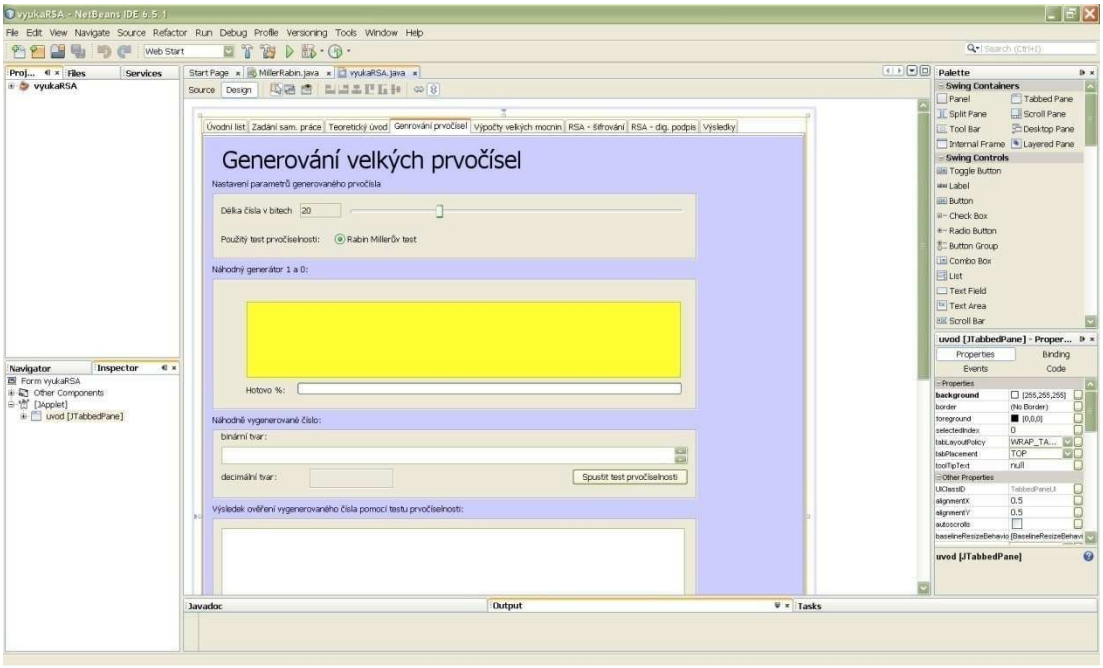
5.2.1 Prostředí NetBeans

Toto vývojové prostředí je open source a volně dostupné. Je k dispozici pro operační systémy z rodiny Windows, Linux, Mac OS i Solaris. NetBeans podporuje následující technologie uvedené v tabulce číslo 2.

Ajax	C/C++	Databases
Debugger	Editor	Groovy
GUI Builder	Java EE	JavaFX
Java ME	Java SE	Java Script
Mobile	PHP	Profiler
Python	Refactor	REST
Rich Client Platform	Ruby	SOA
SOAP	UML	Web
WDSL	XML	

Tabulka 2: Podporované technologie prostředí NetBeans

Pomocí prostředí NetBeans je možné, v případě instalace plné verze, vytvářet plnohodnotné grafické aplikace. Po definici vlastních funkcí, tříd a proměnných automaticky nabízí jejich názvy včetně vstupních a návratových parametrů. Dále dokáže hotové funkce „sbalit“ na jeden řádek, tím zásadním způsobem zvyšuje přehlednost kódu a zefektivňuje programátorskou práci.



Obr. 8: Vývojové prostředí NetBeans

Prostředí NetBeans je dostupné, v případě rozhodnutí nekompilovat si vlastní verzi, v několika balíčcích. Tyto balíčky se liší podporovanými technologiemi. V maximální plné verzi s pluginy umožňujícími plnohodnotný návrh designu programu, zabírá instalace NetBeans cca 242MB.

Z výše uvedených důvodů bylo zvoleno právě vývojové prostředí NetBeans IDE ve verzi 6.5.1. v kombinaci s Java JDK verze 6. Možnou alternativou, nabízejícím obdobné funkce jako NetBeans, by byla volba Eclipse. Rozdíly mezi těmito vývojovými prostředími jsou velice malé. A každý programátor si zastává „svůj“ oblíbený produkt.

6. POPIS SEKCÍ ZHOTOVENÉHO PROGRAMU

Na software sloužící k výuce jsou kladeny specifické požadavky. Dle zadání byl program určen pro studium jednotlivců. Toto primární využití bylo dodrženo a navíc i rozšířeno o možnost práce ve více uživatelích. Tyto možnosti jsou popsány níže. Konkrétně se jedná o sekce šifrování v režimu RSA a sekci digitálního podpisu. K tomuto rozšíření o práci ve skupinách bylo přistoupeno po nastudování metodiky výuky. [4]

Z hlediska co nejlepší přehlednosti výukového programu bylo přistoupeno k možnosti rozdělit celý výukový program na jednotlivé sekce, které budou reprezentovat samostatné tematické okruhy. Celkem je program rozdělen na osm samostatných sekcí. Tři sekce jsou teoretické:

- úvodní list
- zadání samostatné práce
- teoretický úvod

Věnují se popisu programu, zadání samostatné práce a příkladů k vyřešení. Dvě další sekce znázorňují základní úkony prováděné při šifrování a dešifrování pomocí RSA algoritmu.

- generování prvočísel
- výpočty velkých mocnin

Do sekce generování velkých prvočísel je zařazeno i testování prvočísel. Bez znalosti těchto základních požadavků by nemělo studium RSA význam. Původně bylo zvažováno, jestli studentům umožnit případné přeskočení těchto kroků. Po důkladném přezkoumání bylo dosaženo závěru, že výhodnější bude ponechat studentům možnost volby libovolné sekce.

Další dvě záložky v programu se již věnují školnímu využití šifrování a dešifrování pomocí RSA. Dále pak implementaci vytvoření podepsané zprávy a ověření digitálního podpisu zprávy.

- RSA - šifrování
- RSA – digitální podpis

Poslední záložka shrnuje poznatky na základě generovaných hodnot z jednotlivých sekcí. Jedná se o jakousi závěrečnou zprávu.

- výsledky

Celkově je tedy program tematicky dělen. Toto rozdělení přináší nejvyšší možnou přehlednost a efektivitu práce s programem. Umožňují také procvičení nebo pochopení konkrétních nedostatků studenta bez nutnosti se prokousávat celým principem šifrovacího protokolu RSA. Rovněž ovládání jednotlivých sekcí je intuitivní a umožňuje interaktivitu s uživatelem. [4]

6.1 Spuštění programu

Ke spuštění programu není potřeba instalovat žádný software, samozřejmě za předpokladu již nainstalované podpory jazyka Java v internetovém prohlížeči. Poté již stačí spustit soubor vyukaRSA.html. Možnou alternativou by byla nová kompilace a nastavení patřičných parametrů launch.jnlp souboru a následné umístění apletu online.

6.2 Sekce programu „Úvodní list“

Na této záložce jsou informace týkající se této diplomové práce. Na samostatnou výukovou funkci programu nemá sekce vliv. Je zde uvedeno téma diplomové práce, její zadání, jméno vedoucího a autora práce. Další položkou jsou bibliografické údaje, pomocí nichž si může student tuto diplomovou práci vyhledat. Jedná se o úvodní informace, objasňující uživateli obsah a funkce programu a zaměření diplomové práce.

Od uživatele se zde, v této sekci, neočekává žádná interakce a rovněž do závěrečné zprávy se žádné údaje z této záložky automaticky neukládají. Vzhled úvodní sekce programu je v příloze A4 - Obrázek 1.

6.3 Sekce programu „Zadání samostatné práce“

V této části programu jsou shrnuty požadavky a úkoly, které by měl student splnit, pokud má pochopit základní principy algoritmu RSA. K vyčlenění do samostatné sekce bylo přistoupeno z důvodu, aby bylo možné jednoduchým způsobem do zadání zasahovat a operativně měnit dle aktuálních potřeb.

Opět zde není vyžadován žádný zásah studující osoby. Tato sekce již ale ovlivňuje budoucí práci uživatele s programem. Vzhled této záložky je v příloze A4 – Obrázek 2.

6.4 Sekce programu „Teoretický úvod“

Sekce teoretického úvodu je rozdělena do šesti podsekcí. Dělení do podsekcí je prováděno dle stejných pravidel, jakých bylo použito při členění celého výukového programu. Byly tedy vytvořeny podsekce, z nichž každá rozebírá problematiku ze sekce hlavní, případně rozšiřuje množinu uváděných informací.

- RSA – úvod
- RSA – výhody a nevýhody
- klíče v RSA
- generování prvočísel
- RSA – šifrování dat
- RSA – digitální podpis

Po prostudování údajů uvedených ve výše zmíněných sekcích získá student základní představu o problematice RSA. Jedná se o poslední záložku, ve které uživatel nevyvíjí žádnou aktivní činnost, ale naopak se seznamuje s uvedenými skutečnostmi. Vzhled této záložky je rovněž v příloze A4, Obrázek 3.

6.5 Sekce programu „Generování prvočísel“

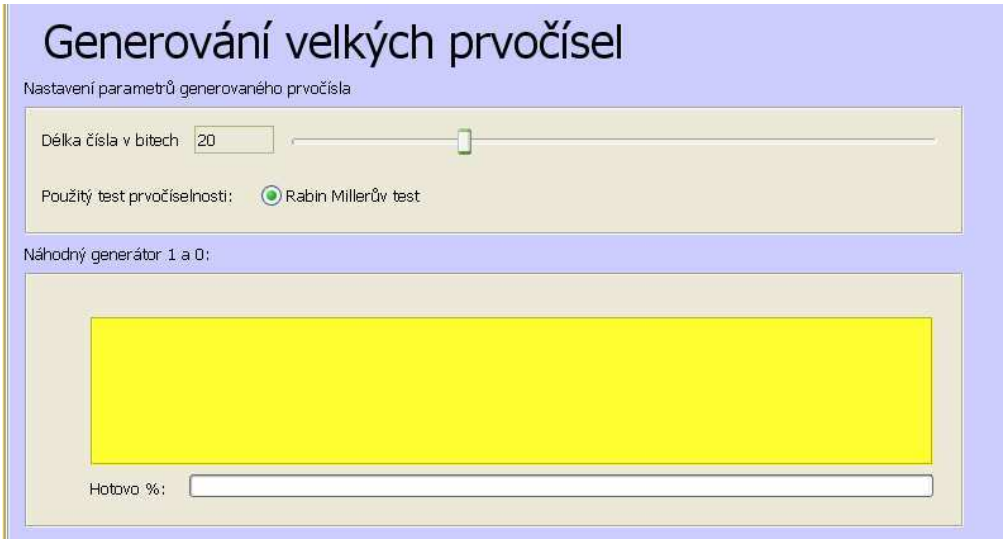
Jedná se o první výukovou sekci. V této části programu si student interaktivně vyzkouší generování náhodné sekvence jedniček a nul, které budou použity pro test prvočíselnosti.

Je použitý jednoduchý princip generování. Na počátek a konec generovaného řetězce jedniček a nul se napevno nastaví hodnota 1. Jedničkou na 1. místě (bit nejvíce vlevo) se zabezpečí požadovaná bitová délka a jedničkou na posledním místě (bit nejvíce vpravo) že se nebude jednat o sudé číslo. Zbytek jedniček a nul je doplněn náhodně metodou náhodného pohybu myši nad žlutou aktivní plochou.

6.5.1 Postup generování

Před započetím generování náhodného čísla, které bude v dalším kroku předáno jako možné prvočíslo k ověření, musí uživatel nastavit délku náhodného čísla v bitech pomocí posuvníku viz.: Obr. 9. Nastavená hodnota se zobrazuje vedle posuvníku na levé straně.

Dalším mezikrokem je uvědomění si použitého testu prvočíselnosti. V programu je v aktuální verzi implementován nejpoužívanější Rabin Millerův test prvočíselnosti.



Obr. 9: Nastavení parametrů generovaného čísla

6.5.1.1 Vygenerování náhodné posloupnosti

Po nastavení bitové délky náhodně generovaného čísla, uživatel může začít pohybovat kurzorem myši nad žlutou aktivní plochou. Na základě pozice kurzoru myši se vrátí její poslední nejméně důležitý (nejvíce proměnný) bit a dle toho jestli se jedná o 1 nebo 0 se do výsledku generovaného čísla započítá 1 nebo 0.

Zároveň se v progress baru, ploše pod žlutým aktivním polem, zobrazuje jaká procentuální část požadovaného náhodného čísla je již vygenerovaná. Jakmile dojde k vygenerování náhodného čísla zadané délky, aktivní pole změní svou barvu ze žluté na šedou. A při dalším pohybu kurzoru již nedojde ke generování dalších jedniček a nul do výsledku.

Tímto získá student náhodnou posloupnost jedniček a nul, které jsou určené k testu prvočíselnosti. Vzhledem k tomu, že studentovi nic neřekne binární posloupnost a navíc je obtížné představit si dekadickou velikost například 40-bitového čísla, byla tato binární posloupnost převedena do dekadické podoby a zobrazena v položce decimální tvar viz.: příloha A4 Obrázek 4. Stále však není vygenerované vlastní prvočíslo.

6.5.1.2 Ověření testem prvočíselnosti

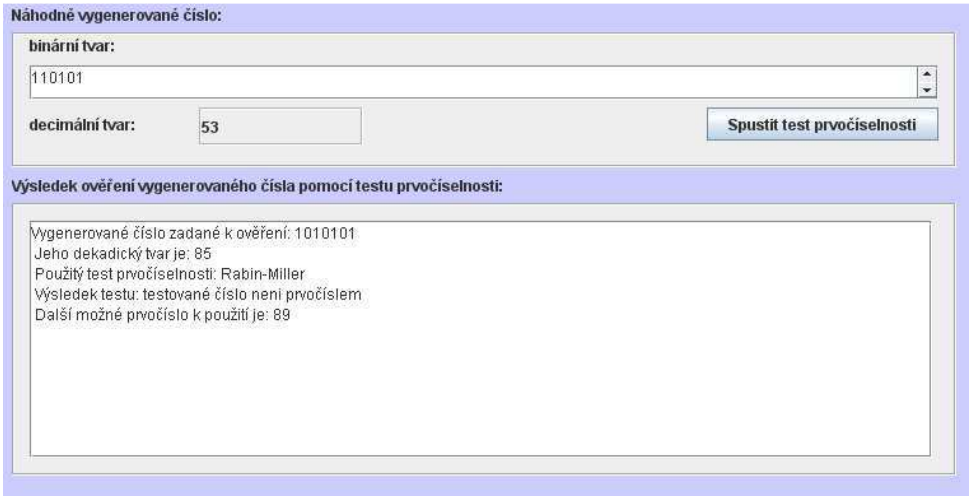
Náhodnou posloupnost získanou z kroku 6.5.1.1 musí student ověřit z některých testů prvočíselnosti. Ve výukovém programu je implementován Rabin-Millerův test prvočíselnosti. Další testy by bylo možné dodat pomocí rozšiřujících tříd. Možnou alternativou by byl například Lehmanův test. Pro princip pochopení generování prvočísel však postačuje jeden test. [2]

Po kliknutí na tlačítko „Spustit test prvočíselnosti“ se zobrazí informace o proběhnutém testu. V případě, že si student vygeneroval opravdu prvočíslo, zobrazí se potvrzení o prvočíselnosti. Pokud ovšem bylo vygenerované číslo číslem složeným, potom pomocí třídy BigInteger, kterou Java standardně obsahuje a její funkce NextProbablePrime() navrženo jiné číslo, které již prvočíslem je. (respektive je prvočíslem s vysokou pravděpodobnosti) a prošlo testem prvočíselnosti.

Tato funkce NextProbablePrime() nepřijímá žádné parametry a je volána pomocí proměnné, ve které je uloženo vygenerované číslo - viz.: níže uvedená ukázka zdrojového kódu.

```
BigInteger n = new BigInteger(text);
    MillerRabin mr = new MillerRabin();
    if(mr.GetRbTest(n)==true)
     {
         // zadane cislo je prvočíslem
     }
    else
     {
       // zadané číslo není prvočíslem a zde je volána poža
       // dovaná funkce
        BigInteger n2 = n.nextProbablePrime();
     }
```

Výsledek ověření tohoto testu prvočíselnosti a další informace, které uživatel získal z této sekce, se zobrazí v textovém poli. Tato dvě vygenerovaná čísla může student použít v sekci šifrování a dešifrování dat. Jak vypadá výsledek a obsažené informace o vygenerovaném čísle je patrný z Obr. 10.



Obr. 10: Výsledek testu prvočíselnosti

6.6 Výpočty velkých mocnin

Druhá sekce programu, která umožňuje vlastní výpočty algoritmu RSA. Protože by kapacita běžných datových typů byla velice rychle vyčerpána, používá se speciálního algoritmu pro výpočty velkých mocnin a jejich modula. Tento algoritmus vychází z rovnice 10. Na počátku je studentovi automaticky vygenerován příklad viz.: Obr. 11. Tento příklad je určen k samostatnému řešení dle zákonitostí uvedených v kapitole 4.4.

Obr. 11: zadání příkladu počítání modula velkých čísel

Jakmile student dojde k výsledku příkladu, který považuje za správný, zadá ho do připraveného políčka výsledku rovnice. Nyní může kliknout na tlačítko „Vyřešit příklad a ověřit výsledek“.

Pomocí tohoto tlačítka bude pro aktuálně zadaný příklad zobrazen postup výpočtu, viz.: příloha A4 Obrázek 5., do textového pole ve spodní části a dále potom doplněna informace o správném výsledku a jeho porovnání s výsledem zadaným studentem. Na Obr. 11 je vidět červeně vypsaná informace o chybně zadaném výsledku příkladu. Pokud by byl zadán korektní výsledek, byla by zobrazena na stejném místě hláška zelenou barvou a textem: „Byl zadán správný výsledek!!!“

Z této sekce si uživatel nepřenáší žádná data dále. Její další využití, pokud není brán v potaz její přínos z hlediska objasnění teorie rozkladu mocnin a jejich využití, je při ověřování „na papír“ u šifrování nebo dešifrování dat. Tam je nutný opakovaný výpočet tohoto typu příkladů. Umožní tedy studentovi velice efektivně zašifrovat i dešifrovat zprávu dle pravidel RSA.

V příloze A4 obrázek 5. je kompletní vzhled zobrazení postupu výpočtu i generovaného příkladu. Po kliknutí na tlačítko generovat nové zadání, bude studentovi zobrazen další příklad na téma výpočtu modula velkých mocnin. Generování příkladů zohledňuje samozřejmě i fakt, že vygenerované číslo, které bude mocněno, musí být menší než modulo.

Exponent tedy nabývá hodnot 3 až 255, modulo, které je počítané, hodnot 11 až 255, číslo vstupující do příkladu hodnot 1 až (modulo - 1). Na konci dojde k vymazání zadaného výsledku z minulého příkladu. Kód, kterým se generuje zadání příkladu, je uveden níže:

```
Random rnd = new Random();
int i;

// vygenerování náhodného exponentu
i = 3+ Math.abs(rnd.nextInt()%252); // rozsah 3 až 255
exponent.setText(Integer.toString(i));

// náhodně zvolené modulo
i = 11+ Math.abs(rnd.nextInt()%244); // rozsah 11 až 255
mod.setText(Integer.toString(i));


// velikost mocněného čísla v intervalu 1-(mod-1)
i = 1+ Math.abs(rnd.nextInt()%(i-2)); // rozsah 1 až mod-1
mocnina.setText(Integer.toString(i));

result.setText("");
```

Tento výše uvedený kód je volán po stisku tlačítka „generovat nové zadání“ viz.: Obr**.** 11. Postup výpočtu je následující:

- vypočítají se mocniny 1,2,4….2*x , kde x<=exponent
- zjistí se, ze kterých mocnin lze počítaný exponent nejefektivněji složit
- celý výraz velké mocniny se rozepíše pomocí složeného exponentu

Možným rozšířením algoritmu by bylo, z důvodu teoretické možnosti opětovného přetečení rozsahu proměnných, násobit výsledná čísla, ne všechny mezi sebou, ale počítat násobek výsledného modula a jeho čísla. Pro praktické pochopení principů výpočtu to ovšem není nutné.

6.7 Sekce programu „RSA – šifrování“

V této části programu již student praktickým způsobem začíná používat šifrovacího algoritmu RSA. Procvičí si výpočet dvojice veřejného a tajného klíče. Zafixuje si informaci, ve kterém případě se používá klíč veřejný a ve kterém klíč tajný.

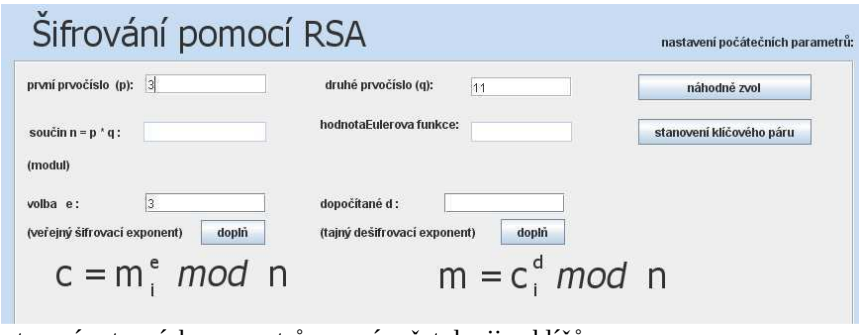
Je to první sekce programu, která umožňuje nejen procvičení principů pro jednotlivce, ale rozšiřuje současně požadované zadání o možnost spolupráce více studentů, čímž opět podporuje uvědomění si principu použití dvojice klíčů a jejich výhody. [4]

6.7.1 Nastavení vstupních parametrů

Prvními kroky, které musí student v této sekci provést, je volba dvou počátečních prvočísel *p, q* viz.: Obr**.** 12. Zde je umožněn výběr z několika možností zadání počátečních prvočísel.

- využití dvou vygenerovaných prvočísel o vhodné bitové délce z kroku generování velkých prvočísel, viz.: kapitola 6.5.
- zadání vlastních dvou prvočísel, ze kterých chceme klíč získat.
- ponechání programem přednastavených hodnot
- využití tlačítka „náhodně zvol“, které vygeneruje dvě prvočísla do příslušných políček

Z důvodu možného ověření výpočtu studentem, který bude provádět kontrolní výpočet na papír, je vhodné nepoužívat příliš velká prvočísla, ale spíš se držet velikosti do 10 bitů.



Obr. 12: Nastavení vstupních parametrů pro výpočet dvojice klíčů

Tímto krokem byly nastaveny počáteční podmínky a je umožněno další pokračování části algoritmu, pomocí něhož získá studující dvojici veřejného a tajného klíče.

Pokračuje stiskem tlačítka „hodnota Eulerovy funkce a součinu prvočísel *p* a *q*“. Tímto se dostává k dalšímu, předposlednímu kroku ustanovení dvojice klíčů (VK,TK – veřejného a tajného klíče). A zbývá již jen stanovení dvojice veřejného a tajného šifrovacího exponentu.

Po stisku tlačítka pod příslušným políčkem veřejného a tajného šifrovacího exponentu bude vhodný exponent doplněn. Tímto krokem získá studující dvojici veřejného a tajného klíče.

VK = dvojice čísel *e, n*
TK = dvojice čísel *d, n*

Nyní jsou stanoveny všechny potřebné parametry, které student potřebuje pro šifrování a dešifrování pomocí tohoto programu. Nyní se způsob prací s programem dělí na dvě možnosti

- samostudium
- práce ve skupině

V případě samostudia odpadá nutnost sdělení (výměny) klíče (klíčů) mezi komunikujícími stranami. Student může začít přímo šifrovat data, pomocí svého veřejného klíče, získat kryptogram *c* a následně tento kryprogram dešifrovat pomocí svého tajného klíče *TK*. V případě, že se rozhodnou studenti této problematiky spolupracovat, musí si vyměnit své veřejné klíče.

6.7.2 Zašifrování zprávy pomocí výukového programu

Parametry nutné k zašifrování pomocí algoritmu RSA byly získány v předchozí kapitole 6.7.1. Nyní si student ***A*** zadá textovou zprávu, která bude zašifrována pomocí zadaného veřejného klíče. Tuto zprávu si může buď vyzkoušet dešifrovat pomocí svého tajného klíče, v případě samostudia nebo poslat nezabezpečeným kanálem svému kolegovi studentovi ***B***, nebo-li majiteli vyměněného veřejného klíče. Tento druhý student ***B*** si obdrženou zprávu dešifruje pomocí svého tajného klíče.

6.7.2.1 Vytvoření zprávy k zašifrování

Student zadá textový řetězec, který bude chtít zašifrovat pomocí algoritmu RSA do kolonky text k zašifrování. Dalším krokem, který je nutné provést je převod textové informace na posloupnost čísel. Ve výukovém programu je to řešeno pomocí převodu textu do ASCII kódu. Převod je uskutečněn pomocí tlačítka převedení textu do ASCII, viz.: Obr. **13**. Pokud má již student vygenerovanou dvojici klíčů, které budou použity pro šifrování. Automaticky mu program doplní hodnoty *e*, *n*. V případě, že se studenti rozhodli pracovat ve skupině, musí student ***A*** zadat veřejný klíč studenta ***B*** a teprve poté může přistoupit k vytvoření kryptogramu.

Obr. 13: převod do ASCII a vytvoření zprávy

6.7.2.2 Omezení pro zadávanou zprávu

Vzhledem k tomu, že bylo pro pochopení principu šifrování vhodnější umožnit studujícímu zvolit menší prvočísla, aby si mohl algoritmus kontrolovat samostatně na papíře a také snadněji spočítat jejich mocniny a uskutečnit převod na kryptogram a zpět, platí pro zadanou zprávu omezení výskytu bílých (netisknutelných) znaků. To z důvodu, že je nutné zprávu zpracovávat dle podmínky uvedené na Obr. 13. Zároveň by ale mělo být umožněno studentovi zadat prvočísla o velikosti třeba 3 a 7. Jejich součin *n* není dostatečně velký, aby mohlo dojít ke zpracování například

po trojčíslí (stovky) nebo po pěticích bitů. Z tohoto důvodu by nebylo možné uskutečnit zpětný převod po dešifrování z ASCII do textové podoby. Toto omezení ovšem nemá vliv na převod do ASCII kódu, následné šifrování a dešifrování. Omezena je pouze funkce zpětného převodu z ASCII do textu.

Druhé omezení, plynoucí z použití ASCII tabulky 0-255, je nemožnost použití znaků s diakritikou. Zadávaná zpráva může být libovolné délky, je pouze nutné počítat s prodloužením doby nutné k šifrování a dešifrování. Jinak pro zprávu nejsou stanovena žádná další omezení.

6.7.2.3 Vlastní zašifrování zprávy

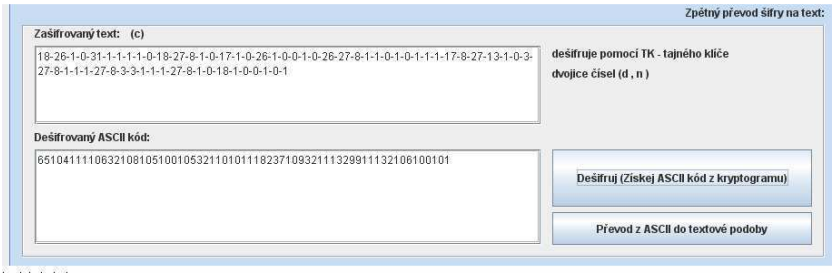
K zašifrování dojde po stisknutí tlačítka „zašifruj“. K šifrování se použije veřejný klíč (dvojice čísel *e*, *n*), zadaný do příslušných vstupních políček viz.: Obr. 13. Převod probíhá po jednotlivých cifrách v dekadické podobě dle vzorce:

$$c_i = m_i^e \ mod \ n$$

Získaný kryptogram se studentovi zobrazí do výstupního pole, viz.: příloha A4 obrázek 6. S tímto obdrženým kryptogramem může student naložit dvojím způsobem. Buď si může, v případě použití svého veřejného klíče, pomocí tlačítka „Získej z ASCII kód z kryptogramu“ dešifrovat zašifrovanou zprávu, anebo pokud použil veřejný klíč studenta ***B***, tak zašifrovanou zprávu odešle pomocí nezabezpečeného kanálu studentovi ***B***.

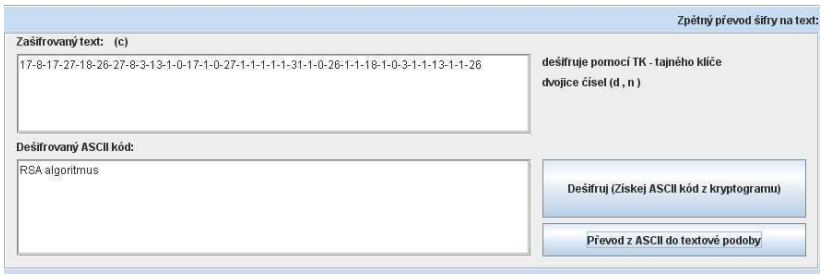
6.7.2.4 Dešifrování obdržené zprávy

Jakmile student získá zašifrovanou zprávu, na Obr. 14 - pole zašifrovaný text, ať již vlastním veřejným klíčem *VK* nebo pomocí komunikace se svým kolegou, může přistoupit k dešifrování zprávy. Tento převod je vlastně zpětně provedené zašifrování. Po stisku tlačítka „dešifruj (získej ASCII z kryptogramu)“ viz.: Obr. 14



Obr. 14: Dešifrování kryptogramu

Tímto student získá ASCII kód, ze kterého byl spočítán kryptogram. Nyní se projeví omezení zadávané zprávy. Pokud původce zprávy dodržel výše uvedené omezení a vyvaroval se při zadávání bílých znaků, pak může student, který zprávu dešifruje, pokračovat stiskem dalšího tlačítka „Převod z ASCII do textové podoby“ viz.: Obr. 15. Po převedení ASCII kódu úloha této sekce programu končí.



Obr. 15: Dešifrování zpětný převod z ASCII do textové podoby

6.8 Sekce programu „RSA – digitální podpis“

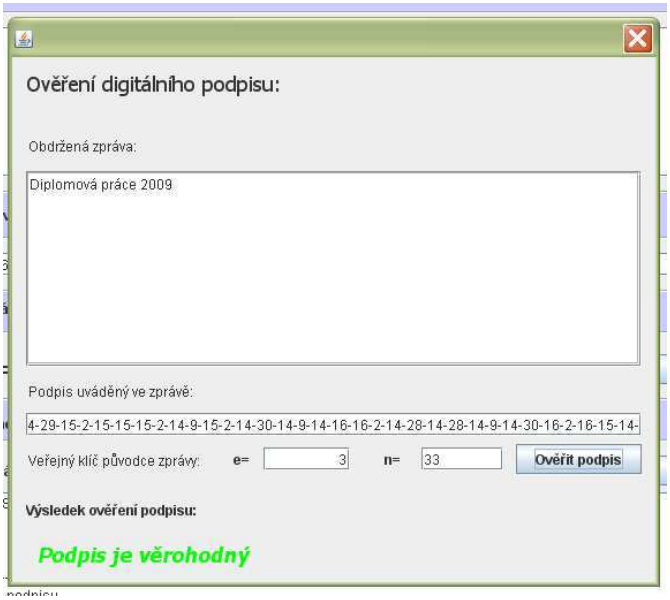
Dalším typické využití RSA algoritmu je v aplikacích digitálního podpisu. V této sekci výukového programu si student vytvoří vlastní textovou informaci. Z této informace získá její hash, který zašifruje. Poté tuto informaci podepíše pomocí svého tajného klíče.

Před započetím vlastního podepsání zprávy je nutné projít sekci šifrování a dešifrování pomocí RSA, kde studující získá tajný a veřejný klíč. Poté již může bez omezení zadat libovolnou textovou zprávu. Dalším krokem je spočítání otisku (hash) zprávy pomocí algoritmu SHA-1. Toho student docílí stiskem příslušného tlačítka, viz.: příloha A4 obrázek 7.

Jakmile je stanoven hash zprávy, je po studentovi vyžadováno načtení tajného klíče, který bude použit pro zašifrování hashe. Tento tajný klíč se načítá právě ze sekce šifrování a dešifrování. Úmyslně není dovolena modifikace nebo zadání ***TK*** *tajného klíče,* aby si student nepletl, který klíč je držen v tajnosti a naopak který je volně zveřejňován.

Po načtení veřejného klíče je dalším uskutečněným krokem provedení podepsání hashe zprávy a připojení vytvořeného podpisu k zadávané zprávě. To student provede kliknutím na tlačítko „vytvořit podepsanou zprávu“. Tím je vytvořena školní verze digitálního podpisu.

Posledním krokem, co student provede, je ověření digitálního podpisu. Po kliknutí na tlačítko: „ověření digitálního podpisu“, automaticky se mu otevře ověřovací okno. Do tohoto okna studující zadá zprávu, kterou chce ověřit, dále potom digitální podpis, který k ní byl připojen a poslední vyžadovanou informací je veřejný klíč podepisující osoby viz.: Obr. 16.



Obr. 16: Ověření digitálního podpisu

6.9 Sekce programu „Výsledky“

Vzhledem k tomu, aby bylo možné umožnit studujícímu znovu projít počítané příklady se stejným nastavením počátečních parametrů a otestovat stejná zadání po jejich prvotním neúspěšném řešení, byla vytvořena tato sekce. Je v ní obsažena závěrečná zpráva. Do této zprávy se ukládají informace pomoci příslušných tlačítek „Uložit do výsledků“ v jednotlivých sekcích. A právě tato závěrečná zpráva umožňuje zpětný návrat k potřebným hodnotám. Vzhled závěrečné zprávy je k nahlédnutí v příloze A4 Obrázek 8.

7. VZOROVÉ ZADÁNÍ A PŘÍKLADY

Šifrovací protokol RSA je v současných aplikacích široce rozšířen. Nachází uplatnění jak při šifrování a dešifrování dat, tak i v aplikaci digitálního podpisu. Seznamte se s principy šifrovací metody RSA, metodou a principem použití dvojice tajného a veřejného klíče. A dále také s požadavky, které jsou na tuto dvojici klíčů kladené.

7.1 Zadání: generování velkých prvočísel

Nastudujte způsob generování velkých prvočísel. Následně vygenerujte několik náhodných čísel a nechejte je ověřit Rabin-Müllerovým testem prvočíselnosti.

Například:

```
Vygenerované číslo zadané k ověření: 1111100111011100011
Jeho dekadický tvar je: 511715
Použitý test prvočíselnosti: Rabin-Miller
Výsledek testu: testované číslo není prvočíslem
Další možné prvočíslo k použití je: 511723

Vygenerované číslo zadané k ověření: 1000100011011111111
Jeho dekadický tvar je: 140159
Použitý test prvočíselnosti: Rabin-Miller
```

```
Výsledek testu: testované číslo je prvočíslem
```

7.2 Zadání: výpočty velkých mocnin

Projděte sekci metody výpočtu velkých mocnin a jejich modula a své teoretické poznatky ověřte v praxi na alespoň několika vygenerovaných příkladech.

Například: $7^{154} mod\ 123 =$?

```
7^1 mod 123=7
7^2 mod 123=(7^1)2 mod 123=7^2 mod 123 = 49 mod 123 = 49
7^4 mod 123=(7^2)2 mod 123=49^2 mod 123 = 2401 mod 123 = 64
7^8 mod 123=(7^4)2 mod 123=64^2 mod 123 = 4096 mod 123 = 37
7^16 mod 123=(7^8)2 mod 123=37^2 mod 123 = 1369 mod 123 = 16
7^32 mod 123=(7^16)2 mod 123=16^2 mod 123 = 256 mod 123 = 10
7^64 mod 123=(7^32)2 mod 123=10^2 mod 123 = 100 mod 123 = 100
7^128 mod 123=(7^64)2 mod 123=100^2 mod 123 = 10000 mod 123 = 37

     Exponent lze slozit z mocnin:
          mocnina: 128 ano
          mocnina: 64 ne
          mocnina: 32 ne
          mocnina: 16 ano
          mocnina: 8 ano
          mocnina: 4 ne
          mocnina: 2 ano
          mocnina: 1 ne
 Celý výraz lze rozepsat na: 7^(2+8+16+128) mod 123 = (49 * 37 * 16
* 37 ) mod 123 = 121
```

7.3 Zadání: šifrování a dešifrování pomocí algoritmu RSA

V dalším kroku vygenerujte dvojici veřejného a tajného klíče. Vygenerování dvojice klíčů provedete pomocí příslušných tlačítek v dané sekci programu. Buď svému kolegovi sdělte Váš veřejný klíč a vyžádejte si veřejný klíč od svého kolegy nebo používejte vlastní dvojici klíčů.

Následně pak tímto klíčem zašifrujte libovolnou textovou informaci. Tu pak předejte svému kolegovi k dešifrování. To stejné udělejte Vy, s kryptogramem obdrženým od kolegy. Prozkoumejte možné modifikace kryptogramu nebo klíčů

a jejich vliv na správnou funkci šifrování a dešifrování. Můžete šifrovat i dešifrovat bez pomoci kolegy, pouze pomocí svých dvojic tajného a veřejného klíče.
Například:

Prvočísla p=139 q=227
Jejich součin n = p * q = 139 * 227 = 31553
Hodnota Eulerovi funkce = (p – 1)*(q - 1) = 31188
Volba *VK* – veřejného klíče, nesoudělného e …… 7
Dopočítaný TK – tajný klíč = 8911

Zpráva určená k zašifrování:

Diplomová práce 2009 – Vychodil

Zpráva převedená do ASCII kódu:

6810511210811110911111822532112114225991013250484857324532861219191041111001051 08

Kryptogram zprávy:

27512-14654-1-0-15019-1-1-128-1-0-14654-1-1-1-1-0-18466-1-1-1-1-1-14654-128-128-15019-2187-128-1-1-128-1-1-16384-128-128-15019-18466-18466-1-0-1-2187-128-15019-0-16384-14654-16384-14654-15019-3165-2187-128-16384-15019-2187-128-14654-27512-1-128-1-18466-18466-1-0-16384-1-1-1-1-0-0-1-0-15019-1-0-14654

Zpětný převod do ASCII:

6810511210811110911111822532112114225991013250484857324532861219191041111001051 08

Zpětný převod do textové podoby z ASCII:

Diplomová práce 2009 – Vychodil

7.4 Zadání: digitální podpis

V dalším kroku vytvořte a následně digitálně podepište textovou zprávu. Tuto zprávu buď ověřte pomocí svého veřejného klíče nebo předejte kolegovi k ověření podpisu. V tomto případě rovněž ověřte Vy zprávu obdrženou od kolegy. Prozkoumejte možné modifikace zprávy, otisku „hash“ zprávy, změnu digitálního podpisu a projevení jejich změn při ověření digitálního podpisu.

Například:

Zpráva k podepsání:

Diplomová práce 2009 – Petr Vychodil

Hash zprávy:

746888f87912c8ebbb124dd18a992a931dfc99b7

Použitý tajný klíč:

TK = **(**7,33)

Podepsaná zpráva:

```
-----BEGIN SIGNED MESSAGE-----
Hash: SHA1

Diplomová práce 2009 – Petr Vychodil
-----BEGIN SIGNATURE-----
Version: školní verze dig. podpisu

     14-14-14-29-14-16-14-30-14-30-14-30-1-0-29-14-30-14-14-14-
     28-16-15-14-0-15-15-14-30-1-0-1-15-2-15-2-15-2-16-15-14-0-
     14-29-1-0-0-1-0-0-16-15-14-30-15-28-14-28-14-28-14-0-15-28-
     14-28-14-1-16-15-1-0-0-1-0-29-15-15-14-28-14-28-15-2-14-14
-----END SIGNATURE-----
```

Ověření validity digitálního podpisu:

Podepsaná zpráva:

Diplomová práce 2009 – Petr Vychodil

Připojený podpis:

```
14-14-14-29-14-16-14-30-14-30-14-30-1-0-29-14-30-14-14-14-
28-16-15-14-0-15-15-14-30-1-0-1-15-2-15-2-15-2-16-15-14-0-
14-29-1-0-0-1-0-0-16-15-14-30-15-28-14-28-14-28-14-0-15-28-
14-28-14-1-16-15-1-0-0-1-0-29-15-15-14-28-14-28-15-2-14-14
```

Veřejný klíč

VK = (3,33)

Výsledek ověření:

Zadaná zpráva nebyla modifikována a podpis je validní.

5) Nezapomeňte si průběžně ukládat získaná data a jako poslední krok projděte automaticky vygenerovanou závěrečnou zprávu v sekci výsledky.

8. ZÁVĚR

V práci byla prostudována problematika kryptosystémů, pracujících na principu faktorizace velkých čísel. V kapitolách 2.1.1 – 2.1.3 jsou navíc uvedeny i základní alternativy kryptografických systémů. Jsou zde pro porovnání a pochopení rozdílu oproti systémům asymetrickým.

Dalším bodem zadání bylo navrhnout koncepci softwaru, který bude sloužit pro podporu výuky. Byl vybrán programovací jazyk Java. Důvody vedoucí pro výběr právě tohoto jazyka a volby výběru vývojového prostředí jsou uvedeny v kapitole 5.

Jednotlivé části výukového programu jsou děleny do částí. Každá z těchto částí je rozebrána v příslušné kapitole. Konkrétně se jedná o kapitoly 6.2 až 6.9. Je v nich uvedeno možné využití výukové sekce při studiu uživatele. Jsou v nich také popsány parametry a vstupní informace, které jsou po studentovi vyžadovány. Dále také výsledky, které jsou v sekci získány a kde je možné těchto hodnot následně v programu využít.

Kapitola **Chyba! Nenalezen zdroj odkazů.** navrhuje možná zadání studijní práce. Vzhledem k tomu, že program je interaktivně ovladatelný a je studujícím umožněno vytvářet vlastní zadání příkladů, bylo upuštěno od pevného zadání a jsou zde uvedeny řešení již vygenerovaných náhodných příkladů, která byla získána právě pomocí programu pro podporu výuky RSA. Studentovi jsou tudíž stanoveny pouze úkoly, které má v dané sekci vyřešit. Konkrétní zadání příkladů tedy získá až v programu po jeho spuštění.

9. POUŽITÁ LITERATURA

1. **SCHNEIER, BRUCE.** *Applied cryptography Second Edition.* New York : Computer Security, 1996. 0-471-11709-9.
2. Zákon č. 227/2000 Sb., o elektronickém podpisu - Část I. - Elektronický podpis. *Bussines center.* [Online] [Cited: 10 10, 2008.] http://business.center.cz/business/pravo/zakony/epodpis/cast1.aspx.
3. **Stallings, William.** *Cryptography and Network Security.* London : Pearson Education, Inc., 2006. 0-13-187316-4.
4. **Skalková, Jarmila.** *Obecná didaktika.* Praha : Grada, 2007. 9788024718217.
5. **Darwin, Ian F.** *Java kuchařka programátora.* Praha : Computer Press, 2006. 80-251-0944-5.

REJSTŘÍK NÁZVŮ

Kryptografie – vědní obor zabývající se šifrováním a dešifrováním dat
Algoritmus – soupis pravidel, podle kterých se zpracovávají data
kryptoanalýza – vědní obor zabývající se prolomením kryptovacích algoritmů
TWINKLE – algoritmus zefektnění luštění RSA
RSA - jsou počáteční písmena autorů této metody
DES – bloková šifra
TripleDES – trojitý (ztrojený) des
AES – advanced encryption standard
Euler – známý matematik
Token – slouží k uchování tajného klíče (k autentizaci)
Interface – rozhraní (počítače, programu)
Modulo – neboli zbytek po celočíselném dělení
kvantový počítač – teoretický stroj
PGP – pretty good privacy – politika klíčů.
Blowfish, - symetrická bloková šifra
GOST, -bloková šifra
IDEA, - bloková šifra
RC2,- bloková šifra
RC5,- bloková šifra
Twofish - bloková šifra
IBM - International Business Machines.
NASA –National Aeronautics and Space Administration
VK – veřejný klíč
TK – tajný klíč
ASCII – způsob kódování znaků

PŘÍLOHA A1

DIFFIE-HELLMANŮV ALGORITMUS

1) POSTUP VÝPOČTU

Je stanovené:

- Velké prvočíslo p
- Generátor g: $y=(g^x) \bmod p$, kde p pro různá x jsou navzájem různá

Odesílatel si zvolí náhodně velké číslo **a**, zároveň příjemce zprávy si zvolí náhodně velké číslo **b**. Tato čísla jsou utajená.

V druhém kroku odesílatel vypočítá $X = g^a \bmod p$ a příjemce $Y = g^b \bmod p$. Tato čísla **X** a **Y** si komunikující strany navzájem vymění.

V posledním kroku si odesílatel vypočítá $K= Y^a \bmod p$ a zároveň příjemce dojde ke stejné hodnotě pomocí $K= X^b \bmod p$. Tato hodnota **K** se nadále používá jako tajný klíč.

Tato metoda využívá skutečnosti, že:

$$\boldsymbol{K = Y^a \bmod p = (g^b)^a \bmod p = (g^a)^b \bmod p = X^b \bmod p} \qquad \textbf{rov. 1}$$

2) PŘÍKLAD VÝPOČTU

Vstupní parametry: Prvočíslo p = 11 Generátor g = 6

Nyní si vypočteme:

$6^x \bmod 11$ pro x náležící1,2,3, …, 10

Tab 1. – vypočtené modulo pro g = 6 a p = 11 z rov. 1

$6^1 \bmod 11 = 6$	$6^9 \bmod 11 = 4$
$6^2 \bmod 11 = 3$	$6^{10} \bmod 11 = 1$
$6^3 \bmod 11 = 7$	
$6^4 \bmod 11 = 9$	
$6^5 \bmod 11 = 10$	
$6^6 \bmod 11 = 5$	
$6^7 \bmod 11 = 8$	
$6^8 \bmod 11 = 2$	

Z výše uvedených hodnot si každý odesílatel i příjemce vybere náhodně jednu hodnotu. Označme si je jako proměnné **a**,**b**.

Postup:
Odesílatel si vybere hodnotu a = 9 a dopočítá si hodnotu X z rovnice 1:

$$X = (6^9) mod\ 11 = 2 \qquad \textbf{rov. 2}$$

Příjemce zprávy si zvolí b = 6 a dopočítá hodnotu Y z rovnice 1:

$$Y = (6^6) mod\ 11 = 5 \qquad \textbf{rov. 3}$$

Z rovnic 1 a 2 se pokračuje ve výpočtu následujícím způsobem:
Obě komunikující strany si nyní vymění své výsledky, které získali z rovnic 2 a 3.
Odesílatel si dopočítá **K** z rovnice 1 pomocí:

$$K = Y^a mod\ p = 5^9\ mod\ 11 = \underline{9} \qquad \textbf{rov. 4}$$

Příjemce si dopočítá **K** z rovnice 1 pomocí:

$$K = X^b mod\ p = 2^6\ mod\ 11 = \underline{9} \qquad \textbf{rov. 5}$$

SHRNUTÍ

Z rovnic 4 a 5 je vidět, že obě dvě komunikující strany došly ke stejnému tajnému klíči K, který je nyní možné používat při synchronní komunikaci. Potenciální útočník není schopen z odposlechnutých hodnot (X,Y) určit zvolená čísla z tabulky 1. (a,b) - princip diskrétního algoritmu. A díky tomu není schopen určit ani tajný klíč K, který se bude následně používat k šifrování komunikace. Bezpečnost této kryptovací metody je srovnatelná například s RSA

PŘÍLOHA A2 – TEST PRVOČÍSELNOSTI

Definice prvočísla:

Prvočísla jsou čísla, která kromě jedničky a sama sebe nemají žádné dělitele. Pokud je potřeba otestovat, zdali dané číslo je opravdu prvočíslem, je dokázáno, že stačí testovat jeho dělitele do velikosti druhé odmocniny z daného čísla.

Možná implementace

```
public static bool prvocislo(ulong cislo)
{
    bool delitelne = false;
    for (ulong i = 2; i <= Math.Floor(Math.Sqrt(cislo)); i++)
    {
        if ((cislo % i) == 0)
        {
            delitelne = true;
            break;
        }
    }
    return !delitelne;
}
```

Popis funkce:

Jako vstupní argument přijímá funkce číslo, které chceme otestovat. Výstupní hodnotou je logická pravda či nepravda. Podle toho jestli je dané číslo opravdu prvočíslem či nikoliv.

PŘÍLOHA A3 Fermatova Faktorizace

Popis algoritmu

Fermatova faktorizační metoda byla poprvé zveřejněna již v 17. Století francouzským právníkem Pierre de Fermatám. Jak již bylo řečeno, je založena na kongruenci čtverců, tedy na vztahu:

$n = x_2 - y_2$, upraveno na

n = (x-y)*(x+y)

Každé kladené liché číslo může být napsáno ve tvaru:

$n = x_2 - y_2$

(Pro naše účely je potřeba, aby $x \neq y$ a také, aby $x-y \neq 1 a\ x+y \neq 1$).

Navíc každé číslo může být napsáno jako n= a * b. Bez újmy na obecnosti uvažujeme **a** > **b** (stejně tak můžeme uvažovat **b** > **a**). Potom můžeme podle předchozího vztahu napsat. [2][1]

$a = x + y$, $b = x - y$.

Po sečtení a odečtení dostaneme vztahy

$a + b = 2x$, $a - b = 2y$.

Po vyjádření **x** a **y** a následných úpravách

$$x = \frac{1}{2}(a + b) \qquad y = \frac{1}{2}(a - b)$$

dostaneme hledaný vztah

$$x^2 - y^2 = \frac{1}{4}[(a + b)^2 - (a - b)^2] = a \cdot b$$

.

Můžeme tedy začít hledat dělitele čísla **n**. Jako první zvolíme

$x_1 = \lceil \sqrt{x} \rceil$, kde funkcí $\lceil x \rceil$ je myšleno nejbližší vyšší celé číslo.

Spočteme $\Delta x_1 = x_1^2 - n$

Pokud je Δx_1 druhá mocnina některého přirozeného čísla, našli jsme hledaná

x a **y** z prvního vztahu $x = x_1$ a $y = \Delta x_1$ Pokud tomu tak není, přiřadíme $x_1 = x_1 + 1$ a postupujeme stejným způsobem. V roce 1920 pak Belgičan Maurice Kraitchik přišel na rychlejší způsob nalezení čísel **x** a **y**, a to za pomocí vztahu

$x^2 \equiv y^2 \pmod{n}$

*Výsledky m*ěř*ení*

Pro testování tohoto algoritmu jsem použil stejná čísla jako pro metodu postupného dělení.

Cifer	Číslo N	Nižší prvoč.	Vyšší prvoč.	Čas [ms]
11	86254148927	18911	4561057	546
12	946327149293	366221	2584033	152
13	9585697475537	1005647	9531871	546
14	57902566197397	5129177	11288861	156
15	187680537102659	7340483	25567873	750
16	6185822575906637	74728037	82777801	31
17	25718328789608557	19682849	1306636493	129890
18	475896319735047317	80174459	5935759663	592093
19	6014825667243535523	1685419877	3568740199	45234
20	10004247321115021451	1685419877	5935759663	169359

Tabulka 3.1 *Fermatova metoda – náhodná čísla s právě dvěma faktory*

Tabulka 3.1 ukazuje, že Fermatova faktorizační metoda je stejně jako metoda postupného dělení časově závislá na velkosti faktorů. Na rozdíl od první metody je tato ovšem silnější na čísla, jejichž dělící prvočísla jsou blízká druhé odmocnině tohoto čísla. Můžeme říct, že metoda je konečná. Dokáže vždy s určitostí nalézt všechny prvočíselné dělitele nebo označit faktorizované číslo za prvočíslo. Časově je ovšem kompletní průchod algoritmem a označení daného čísla za prvočíslo mnohem náročnější než metoda postupného dělení. Proto jsem pro měření zvolil řádově nižší čísla. Pro srovnání je zde i jedno prvočíslo shodné s číslem testovaným v první metodě.

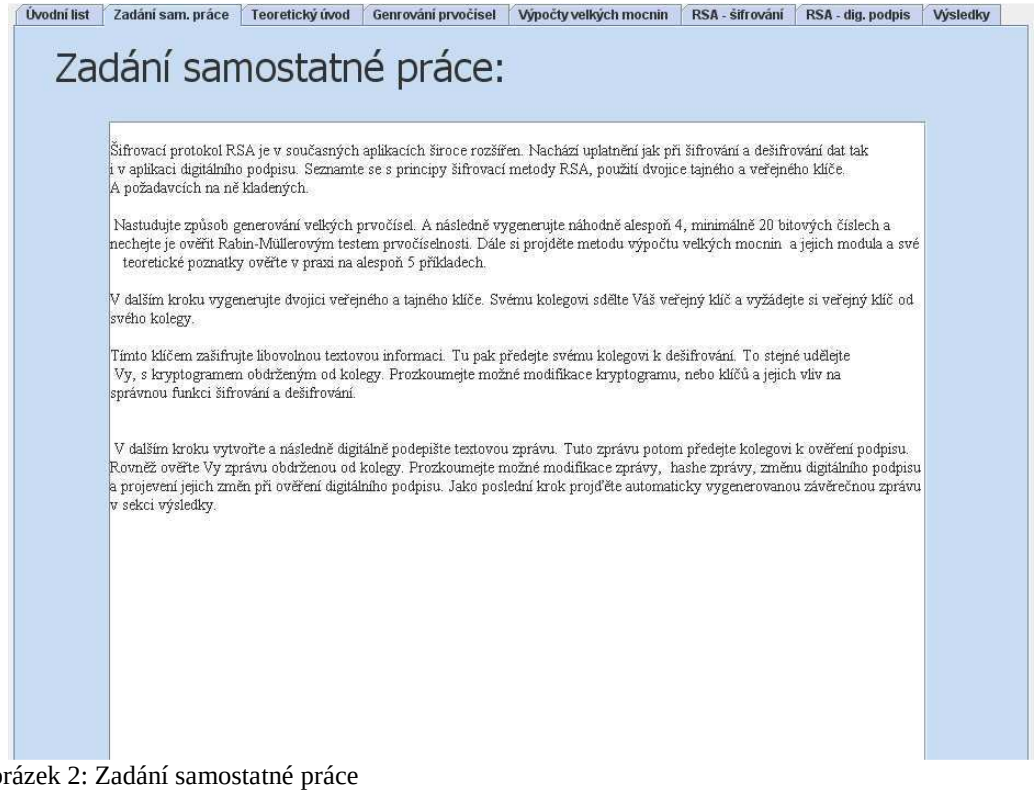
Počet cifer	***Číslo N***	***Čas [ms]***
6	746531	93
7	3589717	500
8	6301479	8921
9	412473419	58984
10	7853025419	1030453

Tabulka 3.2 *Fermatova metoda – prvočísla*

Jak je vidět v tabulce 3.2 již pro deseticiferná čísla přesahuje potřebný čas hranici 15 minut. S použitím metody postupného dělení trvalo určení stejného deseticiferného prvočísla necelou setinu sekundy. Na rozdíl od níže uvedených metod je však spolehlivá. Fermatova metoda je již implementačně i teoreticky složitější než metoda postupného dělení. To však neznamená, že je i rychlejší. Silnou stránkou této metody jsou čísla, jejichž prvočinitelé jsou blízko $\sqrt{n}$. Naopak čísla s malými faktory jsou pro tento algoritmus časově velmi náročná. Vůbec nejnáročnější jsou ovšem prvočísla. Aby algoritmus mohl s určitostí říct, že dané číslo je prvočíslo, musí proběhnout celý algoritmus, což znamená, že s rostoucím faktorizovaným číslem roste i čas. Na rozdíl od následujících metod však můžeme faktorizované číslo za prvočíslo označit jistě. [1][2]

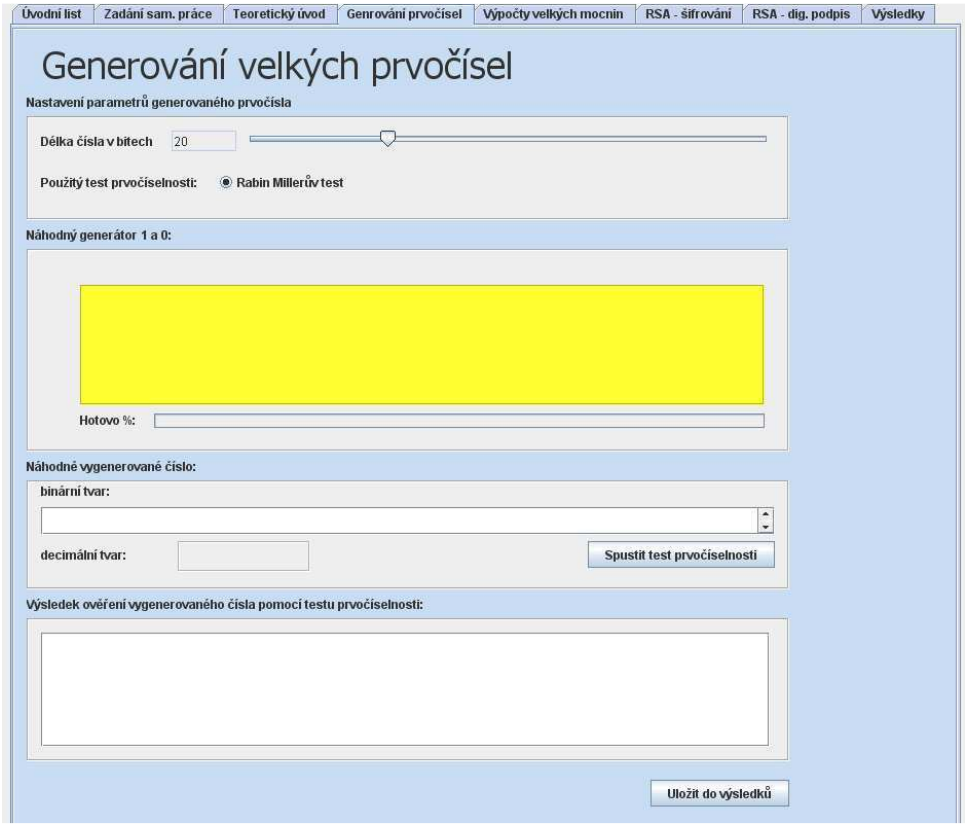
PŘÍLOHA A.4 - ILUSTRAČNÍ OBRÁZKY

Obrázek 1: Vzhled a úvodní sekce programu

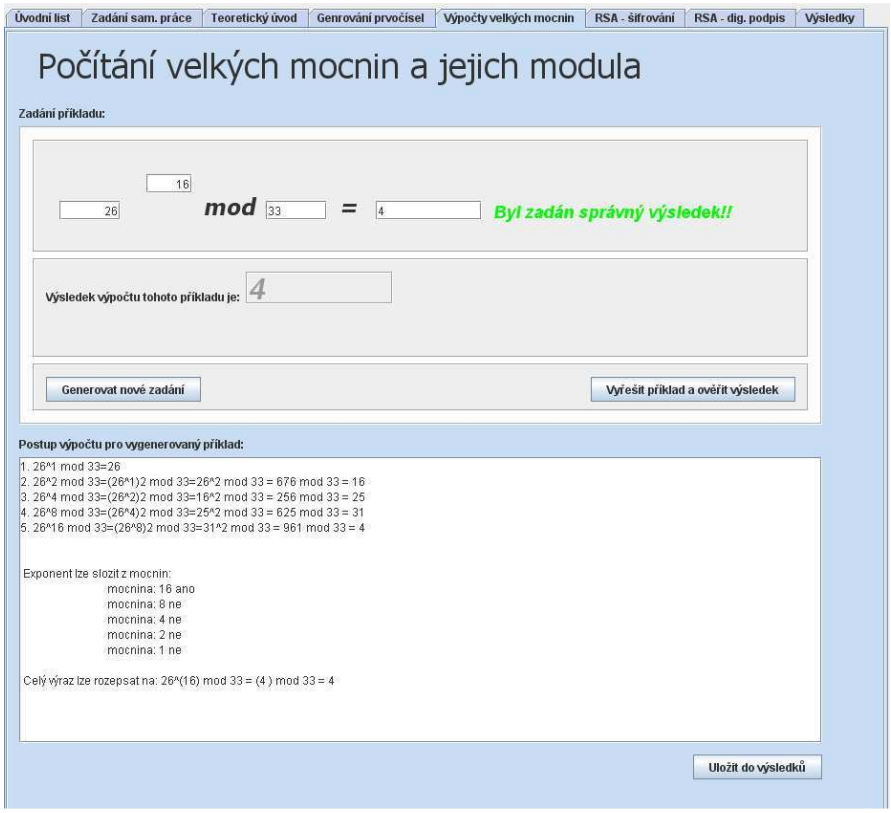


Obrázek 2: Zadání samostatné práce

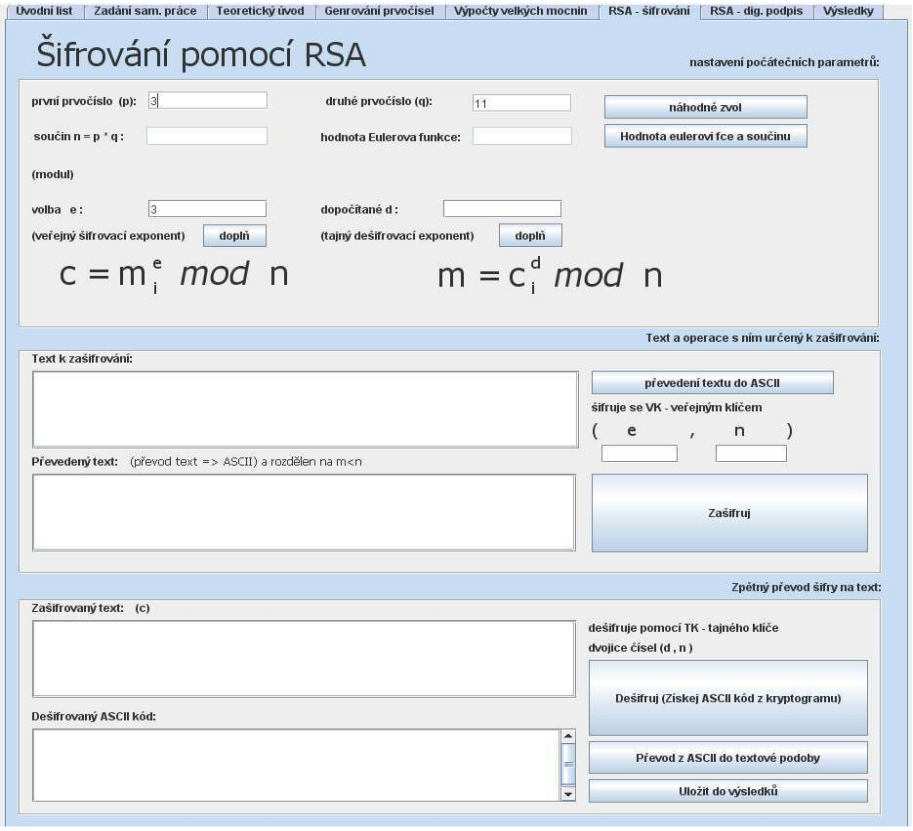
Obrázek 3: Sekce teoretického úvodu



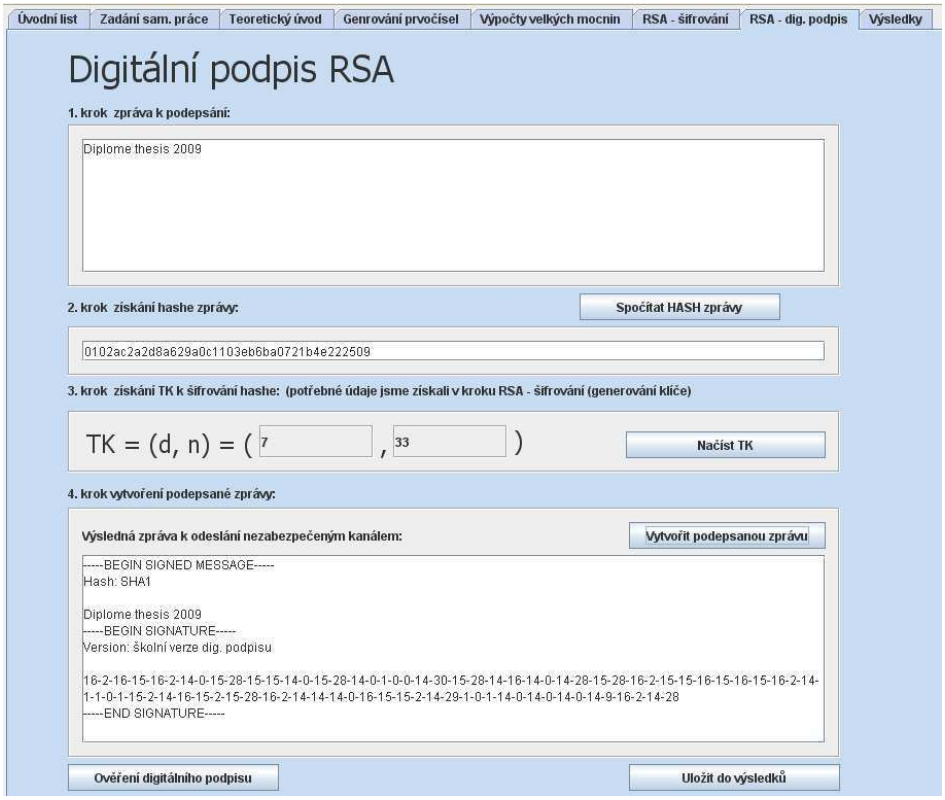
Obrázek 4: generování prvočísel



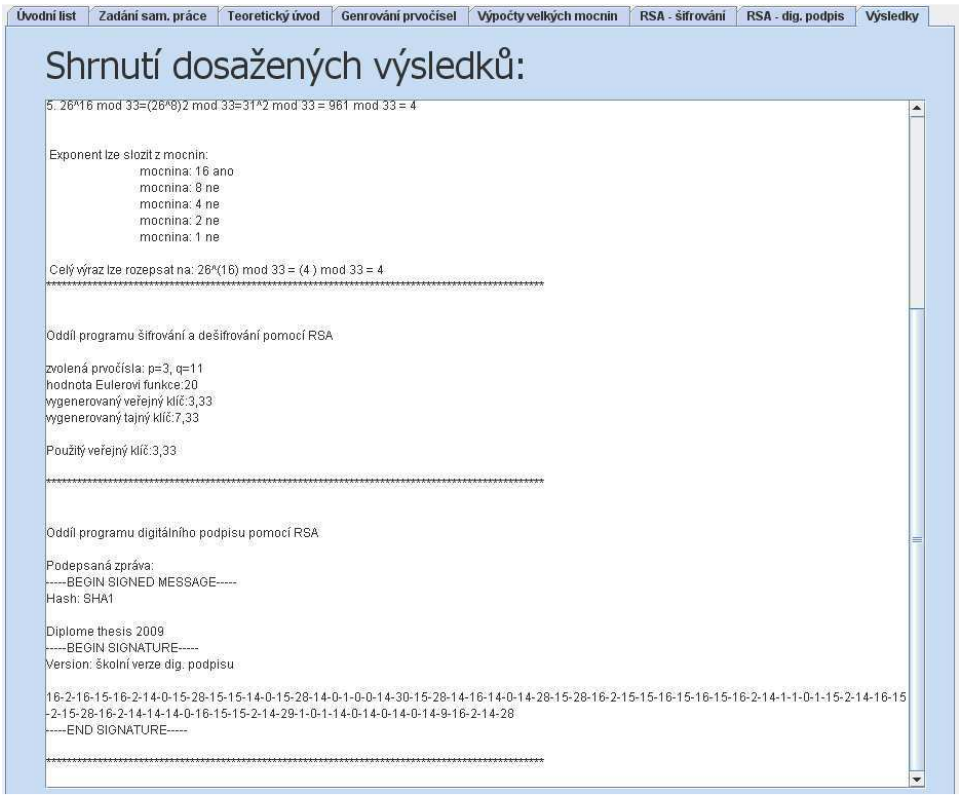
Obrázek 5: počítání velkých mocnin a jejich modula



Obrázek 6: Sekce šifrování a dešifrování dat pomocí RSA



Obrázek 7: podsekce digitálního podpisu



Obrázek 8: vygenerovaná závěrečná zpráva